Symposium OPEN IVR
Application Development Guide

Product release: 4.0
Document release: Standard 1.0
Date: July 1998

© 1998 Northern Telecom
All rights reserved

Printed in the United States of America

Information is subject to change without notice. Northern Telecom reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

Symposium, OPEN IVR, and Nortel are registered trademarks of Northern Telecom in the U.S. and Canada. Meridian 1 is a registered trademark of Northern Telecom. DEC and VT 420 are registered trademarks of Digital Equipment Corporation. HP, LaserJet, and ThinkJet are registered trademarks of Hewlett-Packard Company. Motif is a trademark of Open Software Foundation Inc. UNIX is a registered trademark of Novell, Inc. X Window System and X are trademarks of the Massachusetts Institute of Technology.

Publication history

July 1998 This is the Standard 1.0 issue of the *Symposium OPEN IVR 4.0 Application Development Guide.*

Contents

About this guide **xi**
Purpose of this guide xi
Audience and prerequisites xi
Guide organization xii
Supporting Release 4.0 documentation xiii
Conventions used in this guide xv

1 Introduction **1-1**
Symposium OPEN IVR Control Panel 1-1
Accessing AP software. 1-3
System Console 1-3
Administration tools 1-4
Information Database Editor 1-5
Report tools 1-5
Application tools. 1-6
Prompt Management tools. 1-7
SmartHelp 1-8
Using the pull-down menus 1-9
Graphical user environment 1-10
Using the mouse in the Application Editor 1-10
Buttons, sliders, and entry boxes. 1-11
Navigating with the keyboard 1-12
Safety features on potentially critical functions 1-12
Additional graphical user interface features 1-13
Screen lock 1-13
Exiting from the Symposium OPEN IVR Control Panel 1-15
Restoring the Control Panel. 1-15
AP software auto-restart feature 1-16
Controlling OPEN IVR execution after start-up 1-17

2 Introducing the Application Editor 2-1

Opening the Application Editor 2-2
Creating and selecting an application file 2-4
Opening an existing application 2-4
Using the Application Editor tools 2-6
Cell palette 2-6
Using the drawing board 2-10
Using the control area 2-12
File menu 2-15
Edit menu 2-17
View menu 2-20
Preferences menu 2-24
Placing cells 2-25
Placing a new cell on the drawing board 2-25
Selecting options on the Parameters page 2-26
Connecting cells 2-26
Off-page connectors 2-28
Moving cells using the mouse 2-30
Displaying text on the drawing board 2-34
Insert and remove blank pages 2-36
Select All on Page option 2-38
Finding a cell in your application 2-38
Saving and naming your application 2-39
Printing your application 2-42
Executing your application 2-43
Closing an application 2-44
Exiting the Application Editor 2-45
Application development hints 2-45

3 All about cells 3-1

Components of a cell 3-1
Cell icon 3-1
Parameters page 3-3
START cell 3-5
Modifying DFLT cell parameters 3-5
Establishing application defaults 3-11
Application default fields 3-13
Applying the new default parameters 3-26
Establishing the cleanup handler cell 3-26

4 Concepts of application design 4-1

Application components 4-1
Application basics. 4-2
Example of an application 4-3
Branches 4-4
Buffers overview 4-6
Accessing saved information from calls. 4-9
Databases 4-12
Identification numbers for prompts and messages 4-15
Outdialing. 4-16
Analog Display Services Interface (ADSI) 4-20
ADSI telephone 4-21
Designing an ADSI application 4-22
ADSI (ds) cells 4-24
Application Programming Interface (API) for issuing level 3 ADSI commands 4-27
Constructing an application 4-27
Building applications 4-28
Creating a development directory 4-30
Automatic Speech Recognition using the LISTEN cell. 4-30
Evaluating multiple results with associated confidence levels 4-31
Supporting DTMF functionality. 4-31
Formatting returned results 4-32
Directing rejected responses 4-32
Reporting exceptions 4-33
Logging audited information. 4-33

5 Building a sample application 5-1

Developing a sample application 5-1
Designing the application. 5-2
Call flow for LEAVEMSG 5-5
Selecting prompts 5-8
Create the application with the Application Editor 5-8
Editing the START (Default) cell 5-10
Placing and naming the cells 5-12
Placing and naming the ANSW cell (cell 1) 5-13
Creating and naming the STOR cell (cell 2) 5-13
Placing and naming the remaining cells 5-13
Customizing the cells. 5-14

Saving the application 5-21
Reviewing the application 5-21
Loading and running the application 5-22
Editing the application 5-22
Debugging an application 5-23
Overview 5-23
Invoking the Graphical Debugger 5-25
Controlling the execution of an application 5-27
Modifying buffers 5-31
Quitting the Graphical Debugger 5-34

6 Using message databases 6-1
Message database cell types 6-3
Applications and the current message 6-4
Opening the Information Database Editor 6-5
Creating and naming the database 6-6
Using message databases 6-8
Creating mailboxes 6-8
Adding messages 6-8
Retrieving messages 6-9
Examples of applications that use a message database 6-9
MakeMailbx application 6-11
MsgMenu application 6-13
LeaveMsg application 6-15
RetrievMsg application 6-18

7 Using information databases 7-1
Planning an information database 7-3
Understanding record layouts 7-4
Determining information for database records 7-6
Creating an information database 7-7
Database templates 7-11
Inserting or changing data in records 7-16
Deleting record numbers 7-19
Exiting from the Information Database Editor 7-19
Loading an existing information database 7-19
Using database information through the application 7-20
Locating and extracting data from an information database 7-22
Formats for exporting and importing data 7-30

Performing database functions from the UNIX command line 7-33
Creating a database. 7-33
Exporting database information to an ASCII file 7-34
Importing information from an ASCII file to a database 7-34
Viewing command-line messages 7-37

8 Creating User Functions 8-1

Understanding the User Function process. 8-2
Accessing the User Function from the call flow 8-3
User Function process 8-5
Designing a User Function 8-9
Coding a User Function 8-10
Building the User Function 8-11
Testing your User Function with standalone mode 8-12
Including the User Function in an application. 8-14
usr.c file 8-15
Advanced User Function techniques 8-25
Task Manager 8-25
Timeout 8-28
Customizable user cells 8-29
Customizable USER cell builder 8-30
Buffer Labels dialog box 8-35
Customizing a user cell 8-35
Create and save your cell 8-36
Identify the group for the cell on the Cell Palette 8-40
Creating a help file for the new cell type 8-43

Appendix A: Standard prompts A-1

Appendix B: Customization and internationalization B-1

Modifying resource files and colors B-1
User interface language directory B-2
Items in the UI_LANG directory B-6
Help files B-8
Error log B-8
Language files B-9
SmartHelp B-11
Modifying the uifg.hlp file B-11
Multiple speech recognition vocabularies B-11
Internationalization through the LANG and VINP cells B-13
Creating the mvrv.config file B-18

Glossary **Glossary-1**

Index **Index-1**

About this guide

Purpose of this guide

This manual is directed toward the application developer, the individual who will be using OPEN IVR AP software and the licensed Development software, IVR Developer, to develop and update applications for Nortel's Multimedia Resource Server (MRS) voice-processing systems.

Audience and prerequisites

The application developer does not need to have programming skills, but should feel comfortable using computer screens and working with a menu-driven set of instructions. Although no programming skills are required for developing applications, an application developer with programming experience will be able to take advantage of additional capabilities.

An SCO Development license is required for a Development system, for which the Write/Save functions of the Customizable User Cell Builder are optionally available.

Previous IVR application development experience is an asset.

Guide organization

- Chapter 1, “Introduction,” provides general information about accessing AP software and IVR Developer and the components of the Control Panel.
- Chapter 2, “Introducing the Application Editor,” describes the features of the Application Editor and explains the basics of application development using AP software.
- Chapter 3, “All about cells,” presents an overview of cells and their uses. This chapter provides background information that will help you design your applications.
- Chapter 4, “Concepts of application design,” explains the basics of designing an application and describes the available cells you can use to build your applications.
- Chapter 5, “Building a sample application,” details the process for application development: creating the file, selecting and customizing the cells, saving the file, loading and running the application, and making necessary edits to the application.
- Chapter 6, “Using message databases,” describes the functions and components of message databases.
- Chapter 7, “Using information databases,” describes the functions and components of information databases.
- Chapter 8, “Creating User Functions,” AP software is designed to accommodate as many needs as possible. There are times, however, when you may need a function that is not available within the standard OPEN IVR cells. To support these needs, AP software includes a USER cell that allows you to access functions you can program (in C programming language). This chapter shows you how to use this cell.
- Appendix A, “Standard prompts,” provides a list of the recorded prompts included with AP software.
- Appendix B, “Customization and internationalization,” describes the files necessary to modify the colors that appear on your cell palette, and the information and files required to change the language in which AP software is presented.

Supporting Release 4.0 documentation

Additional information about Symposium OPEN IVR 4.0 is presented in the following Release 4.0 documentation

Document	NT number/ A0 number/ P0 number
Symposium OPEN IVR 4.0 Getting Started Guide	NTVE8201 A0732100 P0881271
Symposium OPEN IVR 4.0 Product Guide	NTVE8202 A0732101 P0881273
Symposium OPEN IVR 4.0 System Administration Guide	NTVE8203 A0732102 P0881274
Symposium OPEN IVR 4.0 Application Development Guide	NTVE8209 A0732108 P0881275
Symposium OPEN IVR 4.0 Cell Catalog	NTVE8208 A0732107 P0881276
Symposium OPEN IVR 4.0 Installation, Upgrade, and Maintenance Guide	NTVE8204 A0732103 P0881277
Symposium OPEN IVR Planning and Engineering Guide	NTVE8205 A0732104 P0881278
Symposium OPEN IVR 4.0 Error Messages Guide	NTVE8206 A0732105 P0881279

Symposium OPEN IVR 4.0 Web Option Guide	NTVE8210 A0732109 P0881281
Symposium OPEN IVR 4.0 SQL Server User Guide	NTVE8216 A0732115 P0881282
Symposium OPEN IVR 4.0 Service Console Interface Reference Manual	NTVE8207 A0732106 P0881284
Symposium OPEN IVR 4.0 VT100 Host Connectivity Configuration and Development Guide	NTVE8221 A0732120 P0881286
Symposium OPEN IVR 4.0 Text-to-Speech Option Guide	NTVE8213 A0732112 P0881292
Symposium OPEN IVR 4.0 Speech Recognition Option Guide	NTVE8212 A0732111 P0881293
Symposium OPEN IVR 4.0 Fax Option Guide	NTVE8211 A0732110 P0881294
Symposium OPEN IVR 4.0 CAS and ISDN Signaling Option Guide	NTVE8214 A0732113 P0881296
Symposium OPEN IVR 4.0 Integration Package for Meridian Link User Guide	P0881297 NTVE8215 A0732114
Symposium OPEN IVR 4.0 Prompt Import Utility Guide	NTVE8218 A0732117 P0881400
Symposium OPEN IVR 4.0 SNA Host Communications Guide	NTVE8227 A0732774 P0881750

Additional information

You may find it helpful to consult the following manuals from other vendors:

- *Informix-SE System Administrator's Guide*
- *Ingres 6.4 Database Administrator's Guide*
- *Oracle7 Server Applications Developer's Guide*
- *Sybase System Administrator's Guide*

Conventions used in this guide

Throughout this guide, several typographic conventions have been used to highlight certain types of information:

- Items that appear on the Symposium OPEN IVR screens are identified in quotes, for example:

  the "Timeout" part of the Parameters window

- Prompts, file and menu names, directories, accounts, and options are shown in a different typeface, for example:

  `/sci>`

  `sci/install` menu

  `/u/nortel/3270/getbalance.act` file

- OPEN IVR buffer names are shown in all uppercase characters, for example:

  the CURRENT MESSAGE buffer

- Field names are shown in italics, for example:

  the *Device* field

- Items you must type are shown in bold in a different typeface, for example:

  type **`tn5250`** at the prompt

- Variables shown in command lines appear in italics, for example:

  the `host_cfg`*n* file, where *n* is a variable representing a board number

- Key names you press are shown in angled brackets, for example:

  the <F1> key

- Keyboards usually have keys named <Return> or <Enter> that perform the same function. For convenience, this guide uses the keyname <Enter> to represent both keynames.

Chapter 1: Introduction

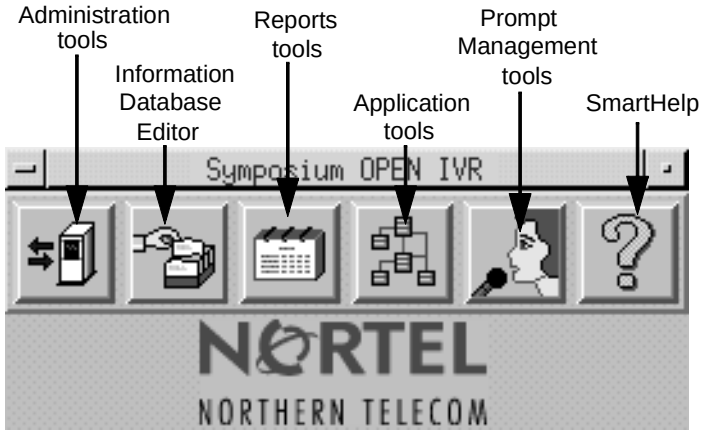
Symposium OPEN IVR AP software is a Nortel software package that allows your company to create its own customized speech applications integrating fax, speech-recognition, and text-to-speech technologies. AP software is one of the few software platforms available that provides robust, easy-to-use speech application development and deployment. AP software runs on the application processor (AP) connected to an MRS such as MRS/AP, MRS/Tower, or MRS/Rack.

This chapter gives an overview of OPEN IVR AP software tools. You access AP software's different tools through the Symposium OPEN IVR Control Panel shown in Figure 1-1.

Symposium OPEN IVR Control Panel

Figure 1-1 shows the Symposium OPEN IVR Control Panel. The buttons available in the Control Panel allow you to access the development and management tools within AP software.

Figure 1-1: Control panel



With the Symposium OPEN IVR Control Panel you can access the following:

- Administration tools
- Information Database Editor
- Report tools
- Application tools, including Application Editor and Customizable User Cell Builder
- Prompt Management tools
- SmartHelp

AP software also contains the following key features:

- graphical development tools
- integrated host communications support
- speech-recognition support
- integrated database management
- multilingual application support
- integrated facsimile support

- graphical prompt editor
- Text-to-Speech

Accessing AP software

To log in to your system, follow this step.

When the system is finished booting, at the SCO Login window, log in as **`user`**.

Upon logging in, OPEN IVR starts automatically as a background process. The System Console and the Symposium OPEN IVR Control Panel come up automatically.

System Console

The System Console provides user information and error messages related to AP software activity. The information that is displayed in the System Console allows you to debug simple system problems and keep track of AP software's overall performance. Figure 1-2 shows the System Console.

Figure 1-2: System Console



The System Console can hold up to 40kbytes of messages in a scrollable buffer. When the console messages exceed 40kbytes, the console removes the earliest messages 4kbytes at a time in order to make room for additional messages. To create a permanent record of selected console messages, you can use the mouse to cut or copy them from the screen and paste them into a file.

The System Console is always active while you are logged in to the AP software account.

See release 4.0 of the *Symposium OPEN IVR Error Messages Guide* for a complete list and description of the System Console messages.

Administration tools



Administration tools provide a menu-driven interface that you use to configure and control your OPEN IVR system.

To access the Administration tools, click mouse button 1 (MB1) on the Administration tools Control Panel button.

The tools available in the Administration menu allow you to

- Configure your system; your AP software application processor can run more than one MRS.
- Assign multiple applications to specific ports, and assign each application to as many telephone trunks as you choose.
- Run multiple applications simultaneously, toggle them, or run them independently of each other.
- Update your applications without losing or interrupting any of your calls.
- Configure MRS ports as inbound or outbound.
- Bring up the System Monitor to monitor specific MRS and port activity.
- Lock your screen to prevent unauthorized access.
- Configure OPEN IVR SNMP Network Management interface.
- Reset OPEN IVR.

Note: In multiple MRS configurations, consult your system configuration to determine which channels are configured on each MRS. Messages must be retrieved separately from each MRS in the configuration.

Information Database Editor



The Information Database Editor is a menu-driven utility within AP software that creates two types of databases:

- a message database that applications use to collect and store messages from callers
- an information database that AP software uses to store data, in the form of character strings, that can be used as part of the application

An AP software message database is used to collect and store callers' speech messages in a set of mailboxes.

To access the Information Database Editor, click MB1 on the Information Database Editor Control Panel button.

You can use an information database to store data (such as names, addresses, and phone numbers of subscribers who use a speech messaging application or customers who use an order entry application).

Each information database consists of a series of records and can be connected to applications or used separately. For example, an application might look up a specific record and then copy information from the record into buffers within the application.

Report tools



Report tools, available from the Control Panel, allows you to access information about system activity. You can find out how many calls AP software has received according to day of the week, hour of the day, or trunk, as well as monitor the events on all or any of the trunks and MRSs controlled by your AP.

Using the Report tools, you can monitor information about system activity. For example, you can do the following:

- Easily find out how many calls have been received by application, cell, day, hour of day, and trunk.
- Track what is happening to the trunks and MRSs controlled by your AP.

- Display and print reports. Display a transaction log with error and informational messages from the AP software processes. You can track what calls were processed through the application cells as well as learn what trunks calls came from and display or print the information you have gathered as reports.

To access the Report tools, click MB1 on the Control Panel Report button.

Application tools



The Application tools menu includes the Application Editor, which enables you to lay out your call flow graphically on a drawing board. When you complete your application, it is an executable file ready to be loaded and run.

The Application tools menu also includes the Customizable User Cell Builder, which enables you to create and modify customized cells (based on the USER cell). You can then use these cells in the Application Editor. With Cell Builder, you can specify a customized icon that you can select from your cell palette and place on the drawing board.

> ***Note:*** You must code the actual functionality for the cell in a separate `usr.c` file. For more information, see Chapter 8, "Creating User Functions," which discusses creating User Functions.

To access Application tools, click MB1 on the Application tools icon in the Control Panel.

The Cell Builder is also invoked when you access the Application Editor.

You can design applications that control the activity on one or more telephone ports connected to an MRS speech system and perform a variety of activities, such as

- Answer a telephone call.
- Play recordings to callers.
- Have callers enter information by pressing designated DTMF key sequences or through speech recognition.
- Have callers make choices by pressing designated keys and have your application carry out these choices.

- Record and store messages from callers and play back these messages to other people at a later time, on demand.
- Make outgoing phone calls designated by the application.
- Access SQL databases.
- Transfer callers to an operator or another line.
- Send and receive faxes.
- Connect callers with external systems (e.g., paging terminals).
- Schedule message delivery applications to be run at designated times.
- Access host databases through 3270, 5250, and VT100 interfaces.
- Give input options through either DTMF or speech recognition as well as play prompts in a choice of different languages.
- Run applications in a choice of different languages.

Prompt Management tools



Prompt Management tools provides several prompt management utilities. The Prompt Loading Facility (PLF) allows you to record your own prompts, load them into the MRS for distribution, and update existing prompts without disrupting your current service.

You can record your prompts directly onto the MRS in three ways:

- with a telephone
- with a microphone
- through a studio-recording tape (for superior sound quality), which is then loaded onto the MRS

If your MRS is connected to an Ethernet network, you can use the PLF to move prompts from the MRS to the AP and distribute them to the other MRSs.

Prompt Management tools also provides access to the graphical System Prompt Editor. This provides a graphical interface in which to edit your prompts.

To access the Prompt Management tools, click MB1 on the Prompt Management tools button from the Control Panel.

Note: Prompt Management tools is a licensed feature.

SmartHelp



SmartHelp provides context-sensitive information about the different Control Panel tools available from the button menus.

To access SmartHelp, follow these steps.

1 With MB1, click the SmartHelp button on the Symposium OPEN IVR Control Panel.

The system displays the Symposium OPEN IVR SmartHelp window.

2 Use the mouse to resize the window, and move it to the side of the desktop so you can access the Control Panel.

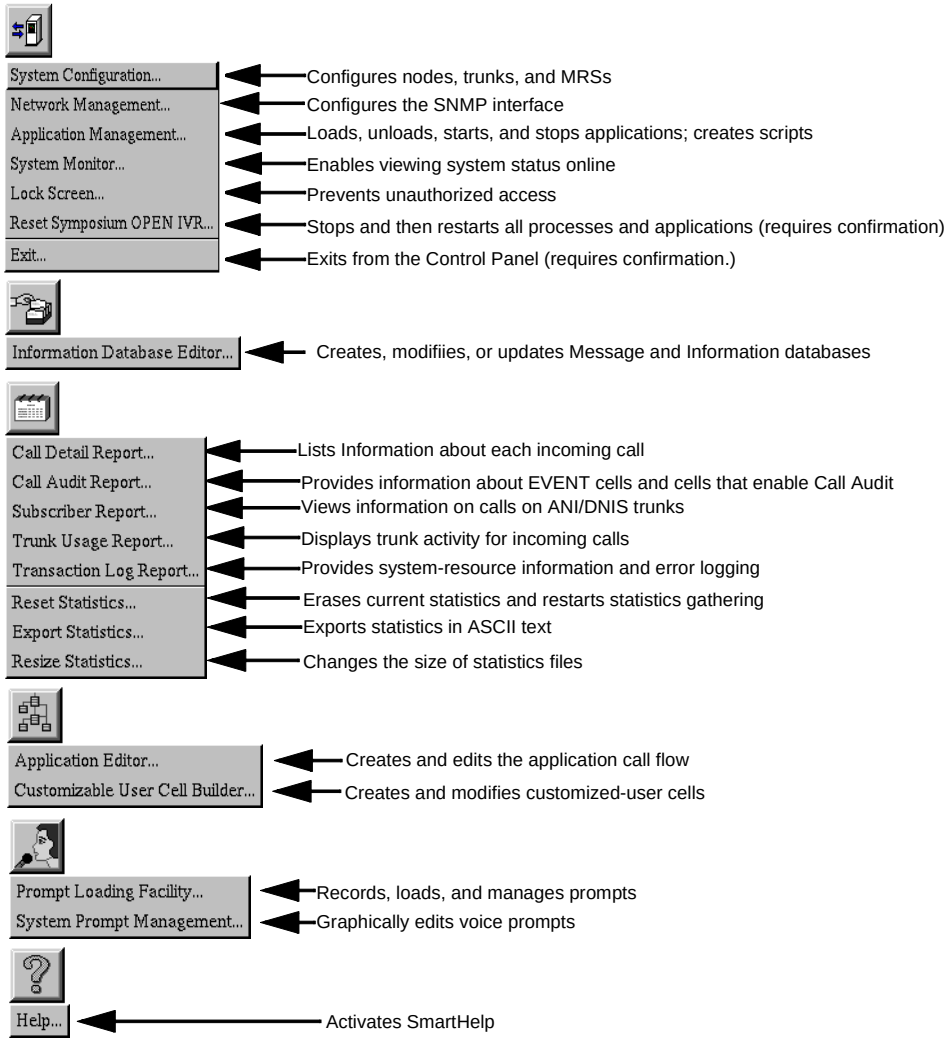
3 As you click other Control Panel buttons and display the menus, information for the appropriate button is displayed in the SmartHelp window.

4 To close the SmartHelp window, click the “Dismiss” button at the bottom of the SmartHelp window.

Using the pull-down menus

Each of the Control Panel options shown in Figure 1-1 displays a pull-down menu. Figure 1-3 contains the different Control Panel buttons with the menu options and a brief description of each.

Figure 1-3: Control panel pull-down menus


System Configuration...
Configures nodes, trunks, and MRSs
Network Management...
Configures the SNMP interface
Application Management...
Loads, unloads, starts, and stops applications; creates scripts
System Monitor...
Enables viewing system status online
Lock Screen...
Prevents unauthorized access
Reset Symposium OPEN IVR...
Stops and then restarts all processes and applications (requires confirmation)
Exit...
Exits from the Control Panel (requires confirmation.)
Information Database Editor...
Creates, modifiies, or updates Message and Information databases
Call Detail Report...
Lists Information about each incoming call
Call Audit Report...
Provides information about EVENT cells and cells that enable Call Audit
Subscriber Report...
Views information on calls on ANI/DNIS trunks
Trunk Usage Report...
Displays trunk activity for incoming calls
Transaction Log Report...
Provides system-resource information and error logging
Reset Statistics...
Erases current statistics and restarts statistics gathering
Export Statistics...
Exports statistics in ASCII text
Resize Statistics...
Changes the size of statistics files
Application Editor...
Creates and edits the application call flow
Customizable User Cell Builder...
Creates and modifies customized-user cells
Prompt Loading Facility...
Records, loads, and manages prompts
System Prompt Management...
Graphically edits voice prompts
Help...
Activates SmartHelp

Graphical user environment

OPEN IVR takes advantage of your APs mouse-driven graphical user interface. The documentation assumes you are familiar with this graphical user interface, the overall operation of your AP, and the SCO UNIX operating system.

Using the mouse in the Application Editor

Your mouse is essential for designing your application. It will serve as a pointer for selecting cells, a pencil for drawing cell interconnections, and scissors and glue for cutting and pasting. Instead of worrying about the mechanics of creating an OPEN IVR application, you can concentrate on the logical design of your application. Table 1-1 lists the Application Editor functions you can perform with a three-button mouse.

Table 1-1: Mouse button functions

Button number	Action	Application Editor function
Button 1 (MB1)	Click	Selects a cell or button; de-select a cell when clicked on background of drawing board area
	Double-click	**On cell:** Opens a parameters window on the selected cell **Off-page connector icon:** Jumps to the page on which a destination cell resides
	Drag	Moves a slider Expands/reduces a window Marks text in a text entry box
	Hold	On scroll bar arrow: Scrolls through a list
Button 2 (MB2)	Click	Places a selected cell on the drawing board
	Drag	Moves a selected cell to another location on the drawing board (can be either the same page or another page).
Button 3 (MB3)		SQL pop-ups in SQL cells. Off-page connector pop-ups.

MB1 typically selects an object to be manipulated, MB2 performs a subsequent action on the selected object, and MB3 allows you to access the SQL cell pop-ups in the Application Editor environment and off-page connector pop-ups. To select a cell from the cell palette, point and click MB1 directly on that object. To select an icon from the cell palette, point and click MB2 directly on the icon. To deselect a selected cell or icon from the drawing board, point and click MB1 on an empty portion of the window.

> ***Note:*** You can use a two-button mouse on SCO UNIX systems by pressing both buttons simultaneously to perform the function of the middle button on a three-button mouse.

Buttons, sliders, and entry boxes

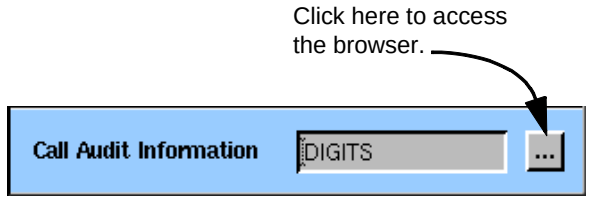
The OPEN IVR interface allows you to take advantage of your APs user-friendly graphical environment. The interface provides Control Panels, push buttons, slide controls, and radio buttons that light up when you turn them on.

Status windows display messages and current selections, text entry boxes allow you to type in options directly, and list boxes let you browse through choices until you find the one you want. Each Control Panel tool takes advantage of these graphical objects in different ways.

Text entry boxes

A text entry box displays the current value for an object (e.g., cell parameter) and is accompanied by an ellipsis push button to access additional valid values from a dialog box called the Browser that appears. For example, the "Call Audit Information" displayed in Figure 1-4 is currently DIGITS.

Figure 1-4: Text entry box with push button



There are two ways you can select a new value. You can position the pointer in the text entry box using one of the actions listed in Table 1-2 and type a new value, or you can select a value from the Browser.

Table 1-2: Manipulating text entry boxes

Action	Process
Select and type over text	Point, click, and drag MB1; type text
Position insertion area	Point and click MB1
Select and type over nearest word	Point and double-click MB1; type text
Replace all text	Point and triple-click MB1; type text

Navigating with the keyboard

You have the choice of manipulating a menu option using either the mouse or a key sequence at your keyboard. You may find it easier to access commonly used options directly from the keyboard using a mnemonic or keyboard accelerator.

Cursor keys

You can also use the cursor keys to move through menu options, and press <Enter> to select the highlighted option, or press <ESC> to cancel the menu without choosing an option.

Safety features on potentially critical functions

Critical functions are protected through various safety features that force you to confirm that you want to perform the action. For example, if you select the `Exit...` option from the Administration button (see Figure 1-5), you must confirm that this is the action you want to perform, as shown in Figure 1-6.

Figure 1-5: Menu option requiring verification

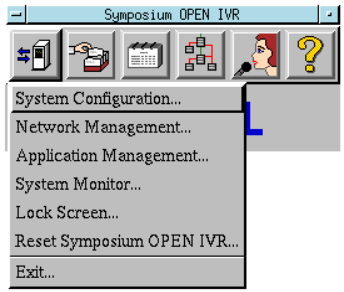
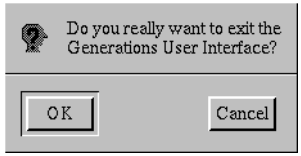


Figure 1-6: Verification pop-up



Additional graphical user interface features

Additional features in the interface allow you to lock your screen as well as stop and restart OPEN IVR.

Screen lock

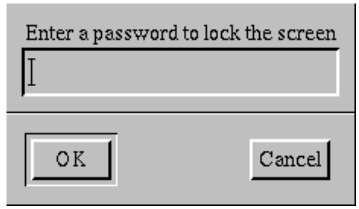
The screen lock feature allows you to lock the screen to prevent anyone from using your terminal while you are not present.

Locking the screen

To lock your screen, follow these steps.

1 Click MB1 on the Administration tools button and select the “Lock Screen” option.

The following window is displayed:



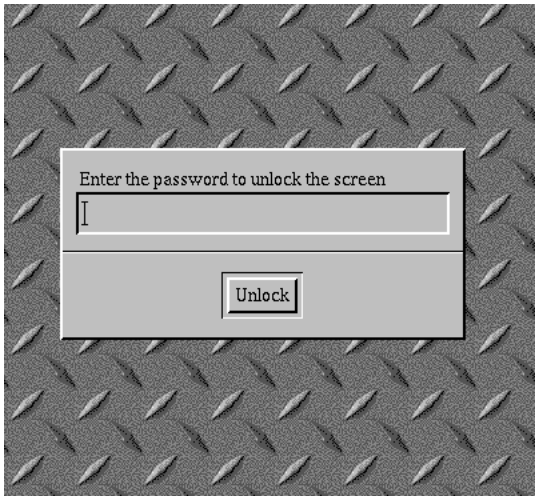
2 Click MB1 on the text entry box, and then type a password.

Note: Only this password will unlock the screen. Select a password that you will easily remember but an unauthorized user cannot easily guess.

3 Click MB1 on the “OK” button to lock the screen (or select “Cancel” to quit without locking the screen).

The system locks the screen and shows a window similar to Figure 1-7.

Figure 1-7: Locked screen



Unlocking the screen

To unlock the screen, follow these steps.

1 Click MB1 on the text entry box, and type the lock screen password.

2 Once you have typed the password, click MB1 on the Unlock button.

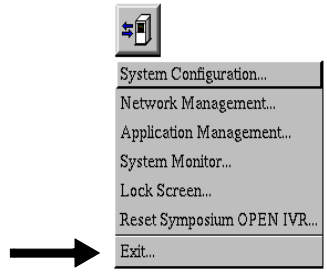
The screen is unlocked, and the desktop is redistilled.

Exiting from the Symposium OPEN IVR Control Panel

Before exiting the Symposium OPEN IVR Control Panel, you should exit all of the windows that you have opened from the Control Panel.

To exit the Control Panel, follow these steps.

1 Click the Administration tools button, and select “Exit” from the menu.

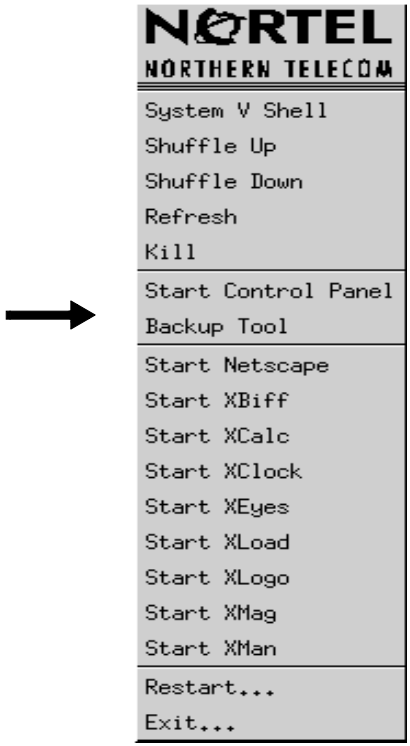


2 Click MB1 on the “OK” button in the confirmation window.

Restoring the Control Panel

You can restore the Control Panel without logging out by using the background menu. If you point on an area of the screen where there are no windows and then hold MB1, the root or background menu will be displayed as shown in Figure 1-8. Although this menu is site-specific, it should include a `Start Control Panel` option. Select this option to restore the Control Panel.

Figure 1-8: Sample background menu



AP software auto-restart feature

Whenever you boot or reboot your AP, the Administration tool's run-time processes are automatically started in the background. If any of the AP software processes are terminated prematurely, AP software is shut down gracefully and restarted automatically by the auto-reboot facility.

During a boot, reboot, or auto-reboot procedure, AP software looks in the `.exe` subdirectory of the AP software's base directory for a file named `reboot.sai`. The `reboot.sai` file is a script that you can create to automatically load applications, set channels, and start applications when AP software is started. This file is useful for initializing the application environment whenever

- the AP is initially booted
- the AP is rebooted following a power failure
- AP software recovers from a software failure
- OPEN IVR has been reset

Initially, there is no `reboot.sai` file. You can create this script file using a text editor or the Application Management "Write a Script" option. For more information, refer to "Creating a Script" in the Release 4.0 *Symposium OPEN IVR System Administration Guide*.

Once the AP boots, you can log in to the account where AP software is installed. The graphical user interface to AP software is automatically started.

Controlling OPEN IVR execution after start-up

If you need to stop, start, or reset OPEN IVR without rebooting the AP, use one of these procedures:

Preferred method

Log in to the account where AP software is installed and select "RESET OPEN IVR" from the Control Panel Administration menu, or type **`greset`** at the shell prompt.

Alternate method

Log in as `root` and type one of these commands depending on the desired action:

Table 1-3: OPEN IVR commands

Command	Action
gen start	Starts OPEN IVR.
gen stop	Stops OPEN IVR and verifies that all OPEN IVR system resources are free. Uses the `genshut` tool if necessary.
gen reset	Resets OPEN IVR.
gen shut	Brings OPEN IVR down forcefully and frees all system resources. You should use this tool only in the case of abnormal system behavior.

Chapter 2: Introducing the Application Editor

By using the Application Editor, you can significantly shorten the time required to develop effective applications. While you are designing your application, the Application Editor simultaneously builds the underlying components and structure. Instead of laying out your design on paper, you do it directly on a drawing board on the screen. All of the design tools you need are right at your fingertips.

After you have opened the Application Editor, you can create a new application or select an existing application file to edit. See “Moving cells using the mouse” on page 2-30 in this chapter for additional information about configuring the cell palette using drag-and-drop functionality. You can also use the OPEN IVR Graphical Application Debugger to debug existing applications. Refer to Chapter 5, “Building a sample application,” for additional information about debugging an application.

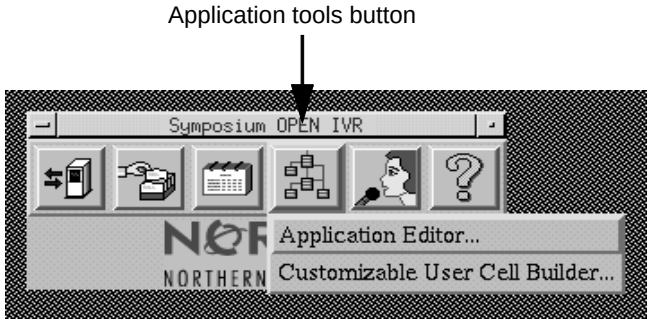
You can draw call flows quickly and efficiently using the mouse. Documenting your design is as simple as saving it to disk or printing it. Online help is available for immediate answers to your questions. You can concentrate on designing the most effective application solution because, as you update a design, the Application Editor automatically builds an executable file.

Opening the Application Editor

To open the Application Editor, follow these steps.

1 From the Symposium OPEN IVR Control Panel, click MB1 on the Application tools button.

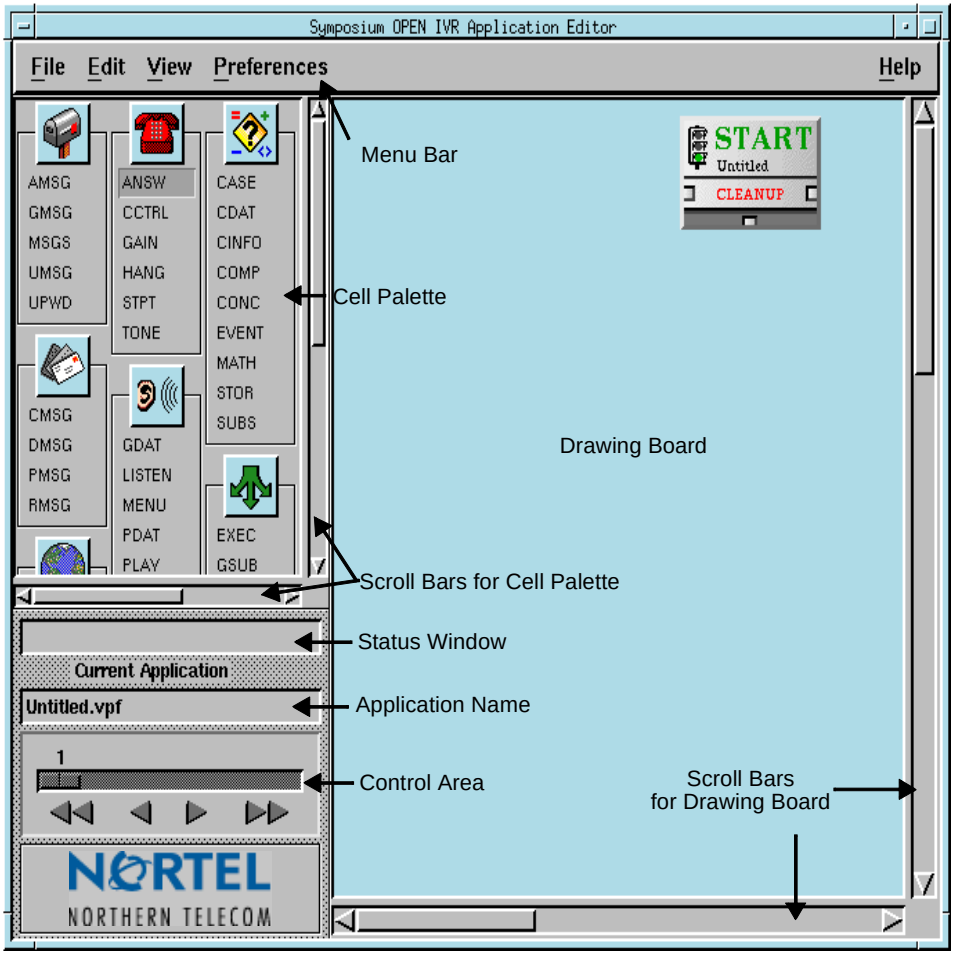
Figure 2-1: Accessing Application Editor from the Control Panel



2 From the menu, select “Application Editor.”

The Application Editor window appears, as shown in Figure 2-2.

Figure 2-2: Application Editor window



The layout of this window consists of four primary graphical tools:

palette	This is your toolbox of available cells, organized by function.
drawing board	This is the area for laying out the design of your application.

control area	The area through which you can move to different pages of the drawing board. It includes push buttons and a slider control.
menu bar	Displays the options for managing your application files (as well as the work you are currently doing on the drawing board).

Once you open an application, a status window displays which application is on the drawing board and which cell has been selected with the mouse.

Creating and selecting an application file

If you want to create a new application, begin your design on the drawing board. If, instead, you want to open an existing application, follow this procedure.

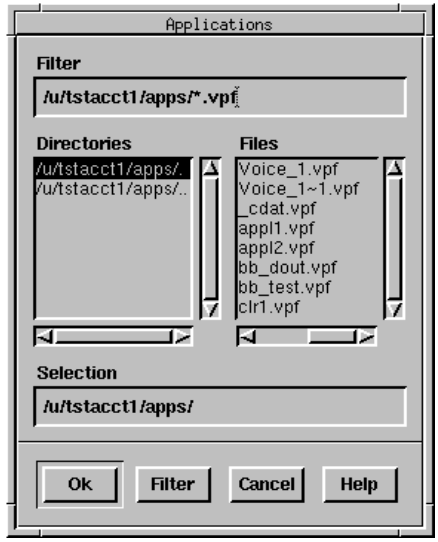
Opening an existing application

To open an existing application, follow this step.

1 Click MB1 on the `File` menu, and, from the menu, select “Open”.

The Editor displays the file browser pop-up window, prompting you to select an existing application, as illustrated in Figure 2-3.

Figure 2-3: File browser



This window permits you to browse through a list of files and then select one, or directly enter a file name.

Table 2-1: File browser fields and buttons

Field/button	Description
Filter	The file browser displays only those file names that contain the character-string pattern that you specify in this field. The default filter is `*.vpf`, which displays all file names in the current directory containing the pattern `.vpf`. To use a different filter, enter your own pattern in the `Filter selection` box and then select the "Filter" push button. Only applications with a `.vpf` filter will run.
Directories	A list of available directories in which applications are stored. You can scroll through these using the scroll bar, and then select a specific directory.

Files	Selecting a file name in the `Files` list displays the name in this box. Otherwise, you can enter a filename directly into this selection box. All the `.vpf` files in the current directory are listed.
Selection	The selected application file. You can retrieve an existing application on your drawing board by typing the application name in the `Selection` text entry box and then pointing and clicking MB1 on the “OK” push button. A shortcut to selecting an application displayed in the browser list is to point and double-click MB1 on the desired application name. The Application Editor redisplays the drawing board with the selected application.
Ok	A push button that opens the application displayed in the `Selection` box.
Filter	A push button that executes the File Filter query and displays a list of application files that match the filter. Pressing <Enter> when the application selection window is open automatically selects the “Filter” push button.
Cancel	A push button that returns control to the Application Editor screen without opening an application.
Help	A push button that displays Help information about this window.

Using the Application Editor tools

Before you begin to use the Application Editor, you should become familiar with how each of the window components can be used, how information and options are presented, and what options are available to manipulate the application you design. This section introduces you to the basic features of the graphical editor and explains what you need to begin designing your applications.

Cell palette

The cell palette contains all of the available cells you can use to develop a voice application. The cells are organized by function, as explained below.



Mailbox

These cells create and access mailboxes, add and retrieve messages, and provide access to mailbox maintenance (e.g., changing passwords): AMSG, GMSG, MSGS, UMSG,UPWD.



Messages

These cells record, play, copy, and delete messages: CMSG, DMSG, PMSG, RMSG.



Prompts and DTMF input

These cells access touch-tone data, present menu selections, play prompts, play formatted data (e.g., numbers, dates, times), and rerecord prompts online: GDAT, LISTEN, MENU, PDAT, PLAY, PNAME, RECP.



Data Manipulation

These cells access buffer information for storing, comparing, or looping: CASE, CDAT, CINFO, COMP, CONC, EVENT, MATH, STOR, SUBS.



Call Management

These cells are used to answer the telephone, generate tones, adjust the gain, provide call control, and hang up: ANSW, CCTRL, GAIN, HANG, STPT, TONE.



Application Scheduling

These cells schedule message delivery, wake-up calls, telemarketing calls, pager notification, and others: DELV, LDLV, TIME, UDLV.



Audio Paths

These cells manipulate the common audio paths when you want to connect to external devices (e.g., music-on-hold applications): CONP, DCNP.



Execute Application

These cells allow one application to invoke another in order to enable modular applications: `EXEC, GSUB, RETN`.



User Process Access

These cells allow OPEN IVR to interface with customer-specific code (for example, proprietary algorithms), access existing MIS applications, communicate with other computers, and retrieve information from files or databases: `USER`.



SQL Server

When the SQL option is installed these cells allow an application to access a local or remote SQL database and execute standard SQL statements to manipulate data: `QCMD, QCNT, QDEL, QINS, QROW, QSEL, QUPD`.



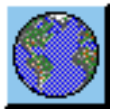
Fax Option

When the Fax option is installed, these cells allow an application to receive and send faxes: `RFAX, SFAX`. Faxes are handled similarly to messages, allowing you to easily integrate speech and fax into the same application.



Speech Recognition

When the speech recognition option is installed, these cells support speech recognition: `SAY, VINP`. You can enable speech recognition in your application and still use the same application to provide DTMF and vocal input.



Multilingual Management

These cells support language-specific prompts: `LANG`. You can select prompts of a specific language to be applied at run time of the application.



Database Management

These cells support the information databases you have created: `CHEK, DDEL, DINS, UPDT`. You can have your application check and update the current database, delete information, or insert new information into the database.



Host Link Management

These cells support host communication support: `COMA, COMI, COMO`. You can have an interface that will allow you to do transactions with a host computer connected to AP software.



ADSI cells

These cells support applications that take advantage of Analog Display Services (ADSI) telephones: `dsCON, dsDATA, dsDIS, dsINFO, dsINPUT, dsKEYS, dsMENU, dsMODE, dsUSER`. You can develop applications that send interactive text-based messages to ADSI-compliant display based telephones.



Outdialing

These cells provide outdialing, bridged outdialing (that is, two trunks: one in and one out), continuous outdial starting with a hook-flash, directed outdial for initiating calls, or dialing digits to a service or pager: `BOUT, CEND, COUT, DIAL, DOUT, XFER`.

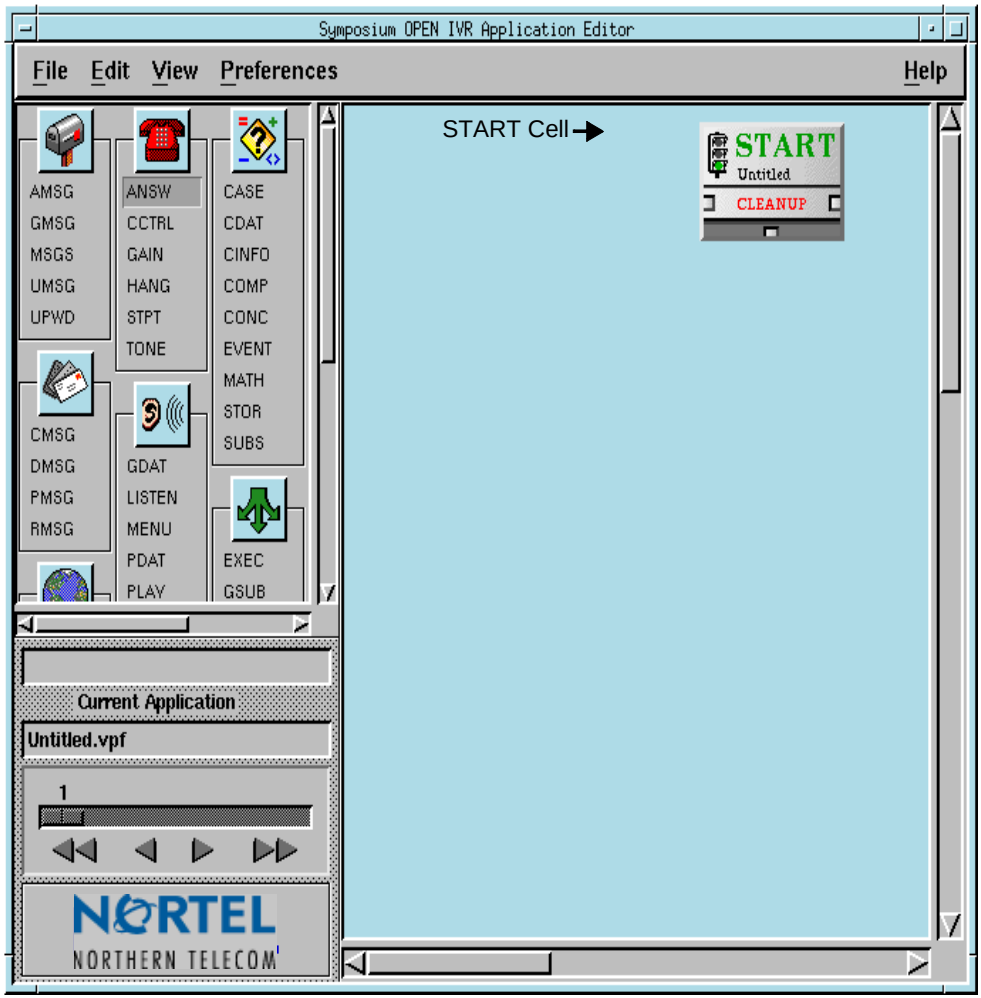
Using the drawing board

The drawing board is the area on the screen where you can design your call flow. As you draw your design, AP software simultaneously constructs the underlying structure of cells and branches.

As shown in Figure 2-4, the drawing board always includes a START (Default) cell, which provides an anchor point from which to begin your design and sets all application defaults. You can use the scroll bars on the right-side and bottom of the window to display other portions of the current page of the drawing board. The drawing board can actually be several pages in length.

> ***Note:*** When you print an application (see “Printing your application” on page 2-42), each drawing board page prints as one 8.5" X 11" page.

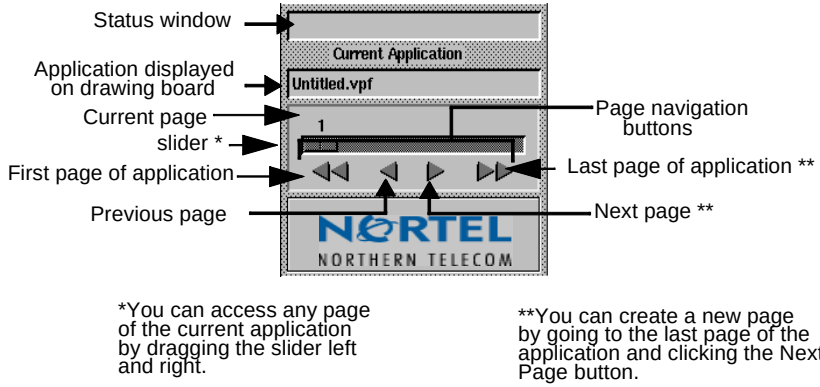
Figure 2-4: The drawing board



Using the control area

The control area of the window includes a status window for identifying which cell is selected, a selection box that lists the name of the current application, and controls for moving to another page of the drawing board. In Figure 2-5, no cell is currently selected on the drawing board, the application displayed on the drawing board is a new application with the default name “Untitled,” and the Application Editor is displaying page 1 of the drawing board.

Figure 2-5: Control area


Status window
Current Application
Application displayed on drawing board
Untitled.vpf
Current page
1
slider *
Page navigation buttons
First page of application
Last page of application **
Previous page
Next page **
NORTEL
NORTHERN TELECOM
*You can access any page of the current application by dragging the slider left and right.
**You can create a new page by going to the last page of the application and clicking the Next Page button.

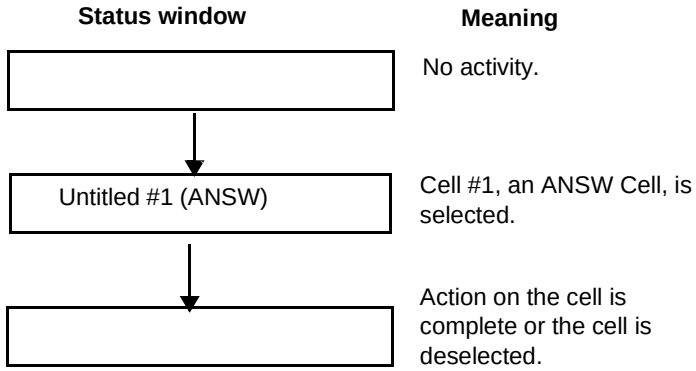
Status window

The status window is your barometer of activity providing immediate feedback as to what is happening with the Application Editor.

When this window is blank, the Application Editor is essentially in a neutral state, waiting for you to initiate an activity. If you point and click mouse button 1 on a cell on the drawing board, the status window indicates that the cell has been selected. This window also displays a message when you load an application, remove a cell from the drawing board, or save the application. As soon as your task is complete, the status window is again blank.

In Figure 2-6, the designer selects a cell for which no name has been assigned. In this case, the status window displays the default name "Untitled." The status window also displays #1, indicating that the cell is the first cell created in the application and that ANSW the cell type.

Figure 2-6: Monitoring activity with the status window


Status window
Meaning
No activity.
Untitled #1 (ANSW)
Cell #1, an ANSW Cell, is selected.
Action on the cell is complete or the cell is deselected.

Note: When no cell is selected on the drawing board (i.e., the status window is blank), MB2 creates a cell at the location of the pointer. When a cell is selected (as noted in the status window), place the pointer at the location where you want the cell and press; MB2 moves the selected cell.

You will find the status window particularly useful when you are working with cells on different pages. For example, if you select a cell on page 1 and then move to page 3 of the drawing board, the control area will still display the name of the selected cell until you complete the task. This information will help you always know when a cell is selected and which one it is.

Current Application

The "Current Application" selection box displays the name of the application on the drawing board. If this is a new application, this field displays the name as "Untitled." If you save the new application under a different name, the selection box displays the new name.

Page navigation

While you are designing an application, your design may expand over one or more pages of the drawing board. The control area clearly displays the current page number. You can manipulate the slider or the push buttons to move to another page.

Creating a new page

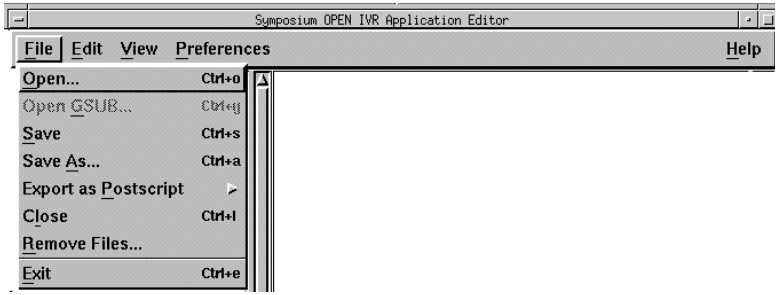
You can easily create new pages. Go to the last page in the application and select the next page button. This creates a new page.

Each page of the drawing board is separate from other pages of the drawing board. This feature provides you with the opportunity to modularize your application. You can place different application functions on separate pages to improve the readability and organization of your design, create a virtually limitless number of application pages, and insert a new page by selecting `Insert Page` from the `Edit` menu.

Using the menu bar

The menu bar shown in Figure 2-7 displays options for managing your application files, as well as the work you are currently doing on the drawing board.

Figure 2-7: The File menu



Each option on the menu bar also includes a mnemonic (displayed as an underscored character in the menu) and an accelerator key (such as <Alt> or <Ctrl> control-key sequence). For example, to open the File menu with the mnemonic, you would press the <alt> key + the <f> key. To execute an item in a pull-down menu, press the appropriate mnemonic key sequence.Menu bar options

Option	Accelerator	Displays a menu of options for
File	<Alt+F>	Manipulating application files
Edit	<Alt+E>	Manipulating data on the drawing board
View	<Alt+V>	Managing default parameters
Preferences	<Alt+P>	Manipulating user preferences

File menu

The `File` menu includes options for manipulating your application files. To access the `File` menu, point and click MB1 on the `File` option on the menu bar.

As shown in Figure 2-7, each option on the menu includes a mnemonic (underscored character) and an accelerator key (such as <Ctrl> or <Alt>) to provide you with greater flexibility in selecting the option. For example, instead of using the mouse to select the `Open` option, you could type <Alt + F> to open the `File` menu, then press <O> with the pointer located anywhere on the window. The following pages list all the accelerator keys used in AP software.

Table 2-2: File menu options

Option	Accelerator	Description
Open	<Ctrl+o>	Displays the file browser window to select an application.
Open GSUB	<Ctrl+g>	Brings up an Application Editor and opens the application referenced by the GSUB Cell. See the description following this table.

Option	Accelerator	Description
Save	<Ctrl+s>	Saves the current application to the current file name. If the current application is new, it is saved as "`Untitled.vpf`."
Save As		Displays the file browser to save the current application under a new name.
Export as PostScript		Creates a PostScript printer file (with "`.ps`" extension) with the same name as the application, which you can later print using a site-specific system command. The submenu allows to choose what you wish to print.
All	<Ctrl+p>	Prints entire application and cell contents.
Call Flow		Prints the graphical cell flow.
Cell Contents		Prints the contents of the cells.
Close	<Ctrl+l>	Closes the current application file and redisplays the initial drawing board. If you have made changes to the current application but have not saved them, the Application Editor prompts you to indicate whether or not you want to close without saving your changes. To save your changes, you must explicitly execute a SAVE or SAVE AS function.
Remove Files		Allows you to remove an application file or any file by changing the filter selection through the file browser window.
Exit	Ctrl-e	Exits the Application Editor.

Open GSUB

To use the Open GSUB option, highlight the GSUB cell in your application, then select this option (or press <Ctrl+G>). If the GSUB cell has a valid application in its Application Name parameter, another Application Editor is displayed and shows the application referenced by the GSUB cell.

> ***Note:*** When you select Open GSUB from the `File` menu, only string literals can be interpreted (for example, the application name "Sample"). The Open GSUB feature cannot interpret the contents of buffer names, such as DIGITS or PATH, although the buffer is a valid entry in the GSUB cell.

Edit menu

You can select the `Edit` pull-down menu shown below to modify the data being manipulated by the current application

Figure 2-8: The Edit menu

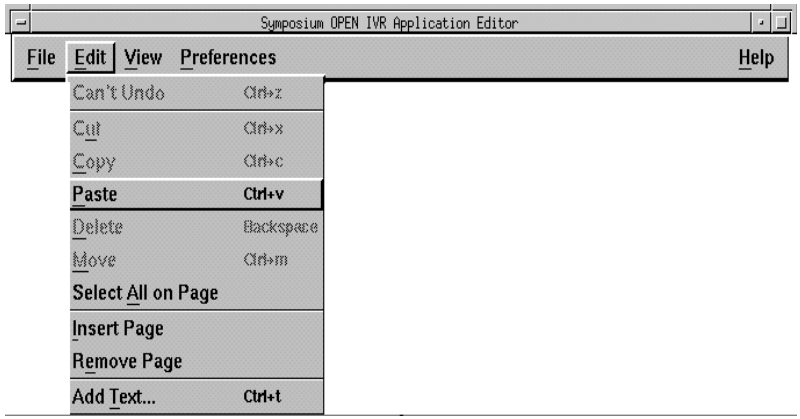


Table 2-3: Edit menu options

Option	Accelerator	Description
Undo (Redo)	<Ctrl+z>	Undoes (that is, removes) the most recent action performed in the drawing-board area. See the description following this table.
Cut	<Ctrl+x>	Removes the selected area from the drawing board to the paste buffer.
Copy	<Ctrl+c>	Copies the selected area to the paste buffer.
Paste	<Ctrl+v>	Pastes the contents of the paste buffer to a new area.
Move	<Ctrl+m>	Moves a selected area while retaining all the current connections.
Insert Page		Inserts a new page before the page selected except before the first page.
Remove Page		Removes a page from the application at any point except the first page. If a page contains cells, then it is not removed.
Delete	<Backspace>	Removes the selected cell from the drawing board.
Add Text	<Ctrl+t>	Adds text to any part of the drawing board in Times Roman font sized at 8, 10, 12, 14, 18, or 24 points.
Select All on Page	<Ctrl+a>	Highlights everything on the page of the drawing board on which the cursor is resting.

Undo/Redo

The Undo/Redo option allows you to undo the last undoable action you took on the drawing board, including the undo itself (redo). Undoing an action automatically creates a `Redo` option. Your levels of undo/redo are limited by only three things:

- Closing the application. Closing an application resets the `Undo` feature. You cannot undo a `Close`.
- Opening the application. Since closing an application resets the Undo feature, you cannot undo actions taken before the application was opened.
- Applying Parameters to a cell. Since the number and complexity of cell parameters would be extremely difficult to track, clicking the "Apply" button in the cell Parameter Page resets the Undo feature.

The `Undo` option under the `Edit` pulldown displays the last undoable or redoable action. The undoable/redoable actions are

- Delete Cell(s)
- Delete Text Item(s)
- Delete Selection
- Create Cell
- Create Text Item
- Cut Cell(s)
- Cut Text Item(s)
- Cut Selection
- Paste Cell(s)
- Paste Text Item(s)
- Paste Selection
- Move Cell(s)
- Move Text Item(s)
- Move Selection

Actions not listed here are not undoable/redoable. For example, If you disconnect two cells and then connect one of those cells to a different cell, you cannot undo that change of connection.

You can switch from undo to redo at any time by clicking on some empty space on the drawing board. For example, if you place five cells on the drawing board, your first undo option is Undo Create Cell. If you select Undo, the fifth cell disappears. Selecting Undo again will undo the placement of the fourth cell. Selecting Undo yet again will undo placement of the third cell. Now, if you want to redo the third cell, you must click on an empty space on the drawing board or else further undos will eliminate the second and first cells. After clicking on an empty space on the drawing board, you have switched from undo to redo. Now selecting undo (redo) will replace the third cell. Selecting redo again will replace the fourth cell, and selecting redo yet again will replace the fifth cell.

View menu

The View menu is shown below.

Figure 2-9: View menu options

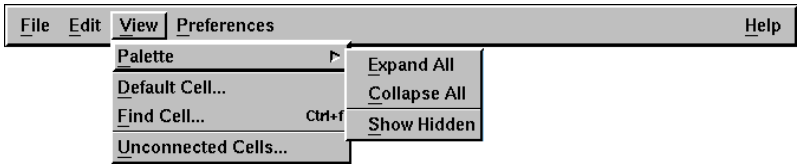


Table 2-4: View menu options

Option	Accelerator	Description
Palette Expand All Collapse All Show Hidden		These options allow you to configure the cell palette to maximize the space available. See the description following this table.
Default Cell		Brings up the Default cell (START) Parameter Page regardless of the application's current page.
Find Cell	Ctrl+F	Allows you to search for a cell in the application by cell number, name, or type.
Unconnected Cells		Brings up a list of cells with unconnected branches in the same way that saving the application does.

Palette

The Palette menu selection has three additional options:

- Expand All
- Collapse All
- Show Hidden

Expanding or collapsing a group of cells

To expand or collapse a group of cells on your palette, follow these steps.

1 From the menu bar, select the View menu.

2 From the View menu, select the Palette option.

3 From the Palette menu, select `Collapse All` or `Expand All`.

A group in its collapsed state does not display any of its cells; a group in the expanded (normal) state displays all cells that are not hidden.

Displaying hidden cells

To display all cells in the collapsed state, (or show cells which have been hidden), follow this step.

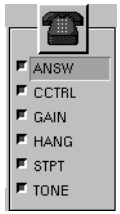
1 From the View menu, select the `Show Hidden` option. This option is shown on page 2-22.

You can use the `Show Hidden` option to hide cells on your palette that you do not use in your application. By using the `Show Hidden` option, you conserve space on your palette by displaying only the cells that you use.

For instance, a typical cell palette icon looks like the following example:



When `Show Hidden` is switched on, all of the cell names on the cell palette will have check boxes next to them as shown below.



Click on the check boxes (putting them in the “unclicked” position) of the cells you wish to remove from the palette.



When you toggle off the `Show Hidden` option, the cell names that have their check boxes in the “unclicked” position will disappear.



Note: Hiding all cells in a group will hide the group’s icon as well.

- To restore a cell name to the cell palette, switch the `Show Hidden` feature on, click the check box next to the cell name, then switch the `Show Hidden` feature off.
- To expand all of the icons on the cell palette, select `Expand All.` This option displays all non-hidden cells under each cell palette icon.
- To collapse all of the icons on the cell palette, select `Collapse All`. This option folds all of the cell names under their respective icons.In addition, you can organize your palette by selecting groups of cells and dragging and dropping the group anywhere on the palette by following this process:

1 To initiate a group drag, click and hold MB2 over the cell group you want to move (this will invert the group graphics, making the group appear depressed). The cursor will change to a hand to indicate that you can drag the group to another location on the cell palette. The palette will temporarily resize to become wider and taller by one group, in order to allow you to extend the palette in either direction.

2 Drag the group to the desired location, and press MB2 again. The cursor will revert to the down arrow, and the group graphic will appear at the arrow’s location (within the confines of the call palette).

Since the cell palette is larger than the window in which it is displayed, you can scroll the palette by sliding the arrow off the window in the direction in which you wish to move.

You can cancel the drag at any time by depressing a different mouse button, or releasing MB2 outside of the valid drop area.

Preferences menu

The Preferences menu allows you to configure some of the global characteristics of the Application Editor

Figure 2-10: The Preferences menu.

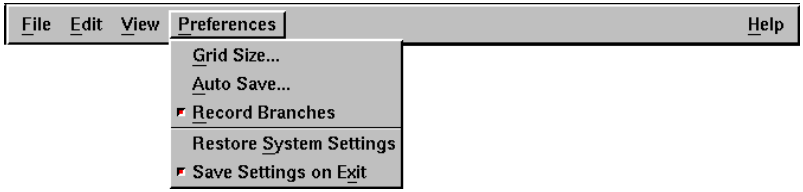


Table 2-5: Preferences menu options

Option	Description
Grid Size	Displays the Adjust Grid Size pop-up window to modify the coordinate resolution (pixels) of new cells placed on the drawing board.
Auto Save	Allows you to customize your auto-save feature for backing up the application automatically or cancels the operation.
Record Branches	Allows you to select multiple output branches to connect to one input branch.
Restore System Settings	Restores the cell palette settings to their original factory-shipped state when you exit the Application Editor. This feature overrides the “Save Settings on Exit” option.
Save Settings on Exit	Saves your current custom cell palette settings when you exit the Application Editor. The next time you bring up an Application Editor, it will have your custom cell palette.

Placing cells

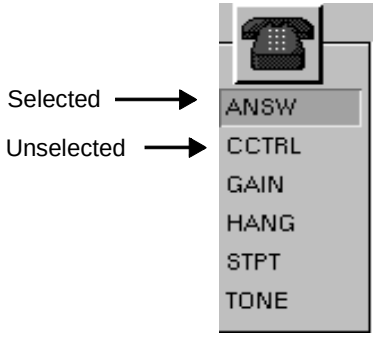
Placing a new cell on the drawing board is easy. Perform the following steps.

Placing a new cell on the drawing board

To place a new cell on the drawing board, follow these steps.

1 On the palette, click MB1 on the name of the cell you want to place.

This selects the cell. The following figure shows “ANSW” as the selected cell.



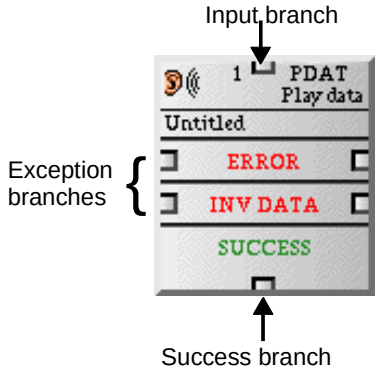
2 Click MB2 on an empty area of the drawing board where you want to place the cell.

The Application Editor displays a cell icon for the selected cell. When you perform this step, make sure you do not point and click on a cell already located on the drawing board and make sure that a cell is not already selected (check the status display), since the Application Editor will interpret your action as a move-cell operation (see “Moving a Cell on the Drawing Board”).

Note: As an alternative to positioning the pointer on the drawing board before “dropping” the new cell, use this technique: hold MB2 until an outline of the cell appears, then drag MB2 to where you want to place the cell, and release the button.

Each cell that you place on the drawing board appears as an icon with the same basic features as the PDAT cell shown in Figure 2-10. Other cells will have different branches depending on their function.

Figure 2-10: PDAT cell



Selecting options on the Parameters page

To simplify your design effort, the Application Editor provides a default set of values for the parameters used by the cells in your application. To view these defaults, double-click MB1 on the cell icon. The cell's Parameters page will appear. You can edit items on this page by typing in your selection or by designating information by means of a slide bar, radio buttons, browsers, and DTMF digit push-button option menus, depending on the option. Chapter 3, "All about cells," goes into detail on configuring Parameters pages.

Connecting cells

You connect cells by connecting an exception or success branch of one cell to the input branch of the next. Click MB1 on one of the cell branches. The mouse pointer turns into the pencil cursor shown below.



Click a branch of another cell. A line connecting the branches will appear. Success branches are solid lines, and exception branches are dashed lines. Figure 2-11 shows connected cells.

Figure 2-11: Using the pencil cursor to connect cells

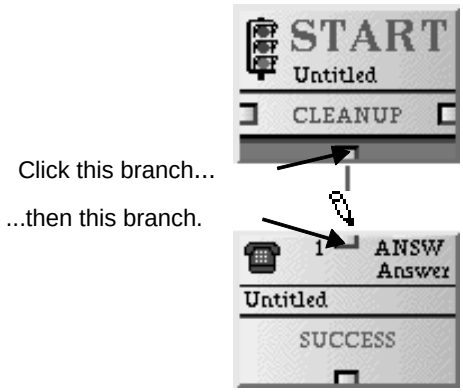


Figure 2-12 illustrates both success and exception branches.

Figure 2-12: Connected cells



A shortcut to connecting cells is to create the first cell, point and click MB1 on the cell's branch connector, then click MB2 at the appropriate location on the drawing-board area. Because you activate the branch connect tool before creating the second cell, the Application Editor automatically generates the branch when you create the cell. This technique is particularly useful when you are creating or connecting cells between pages.

To connect multiple output connectors to a single input connector, follow these steps.

1. Make sure the Record Branches option from the Preferences menu is on.
2. Click MB1 on each of the output connectors you want.
3. Click MB1 on the input connector to which each of the selected output (source) connectors will be connected.
4. The pencil tool will be deactivated, and branches from each source to the destination connector will be displayed.

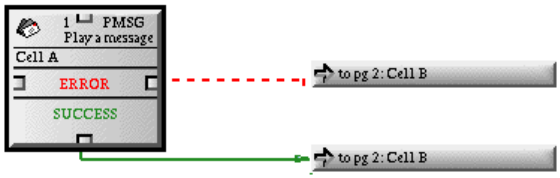
Off-page connectors

It is likely that your applications span multiple pages. In this case, you use off-page connectors to connect cells placed on different pages. To connect Cell A on the current page to Cell B on another page, follow these steps.

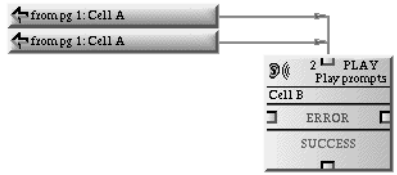
1. Point and click MB1 on an output connector (that is, a success or exception branch) of Cell A on the current page.
2. The pointer changes to the pencil cursor.
3. Navigate to the page containing Cell B.
4. Click MB1 on the input connector of Cell B.

Figure 2-13 illustrates an off-page connector indicating a success branch from the “Play a Message” (PMSG) cell to another cell on page 2 of the drawing board. Figure 2-13 also shows the corresponding off-page connector from page 2 of the drawing board to the “Play a Message” (PMSG) cell.

Figure 2-13: Identifying a branch from a cell on another page



Off-page connector branching to a cell on another page



If you double-click MB1 on an off-page connector, the Application Editor goes directly to the indicated page and selects the destination cell so you can find it quickly. For example, double-clicking the off-page connector in the second illustration in Figure 2-13 shows page 1 with the PMSG cell selected.

After you have one exception and one success off-page connector connected to an input branch, additional off-page connectors connecting to that input branch will stack themselves on the existing connectors. To see where all the stacked off-page connectors lead, click MB3 on the connector. A menu will appear displaying all of the cells that connect their success or exception branch to this cell. You can select a cell from this pulldown, and AP software will automatically navigate to that cell and highlight it.

> ***Note:*** Once an off-page connector exists on the page you are working on, you can easily create another off-page branch to the same cell without changing pages. Click MB1 on the output connector of the cell on the current page and on the arrow of the off-page connector. The branch will automatically be created between the two cells.

Viewing multiple off-page connectors at the same location

If you have more than one off-page connector on the current page and they are positioned at the same location on the drawing board, a one-pixel-wide red box is displayed around the first connector to indicate the existence of more than one connector. You can click MB3 on this connector to access a pop-up menu that lists the off-page connectors residing at that location. Selecting one of the entries in the list positions the cell being pointed to in the center of the drawing board.

The pop-up menu displays up to 10 entries at one time. If there are more than 10 onto-page connectors, click MB3 to display the first 10 entries, and then again within 2 seconds to display the next 10 entries, and so forth. Once the end of the list is reached, clicking MB3 again takes you back to the beginning of the list.

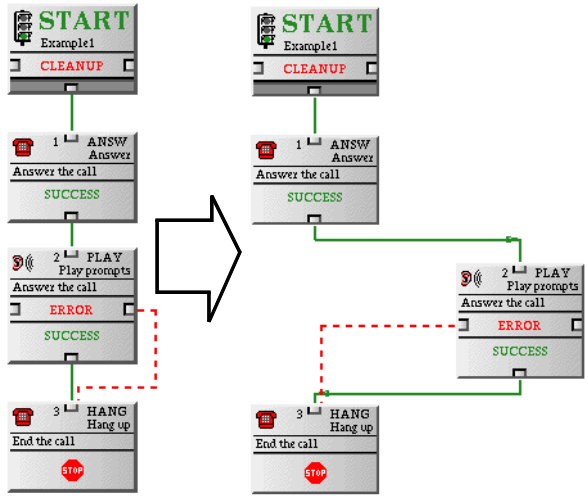
Moving cells using the mouse

Whether you want to move the cell to a different location on the same page or to another page altogether, the process is simple. This section describes how to move a cell using the mouse, and illustrates how the cell connections move with the cell. You can also move one or more cells by using the Move option on the Edit menu. Refer to the sections on using the Cut, Copy, Paste, and Move options and using the Edit menu options later in this chapter. When you move a cell, it maintains all of its connections.

Moving a cell on the same page

To move a cell, click MB2 within the cell and drag it to the new location. Or, click MB1 in the cell. Then move the cursor to the new selection and click MB2. Any branches associated with the cell are automatically resized to maintain the connection. Figure 2-14 illustrates moving a PLAY cell.

Figure 2-14: Sample move of a cell



Before move **After move**

Moving a cell to another page

To move a cell to another page, follow these steps.

1 Point and click MB1 on the cell to be moved.

The status window indicates that the cell is selected.

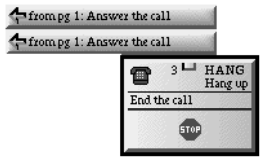
2 To access the page to which you want to move the cell, use the page selector push buttons or slider. Notice that the status window indicates that the cell is still selected.

3 Click MB2 at the location on the drawing board where you want to place the cell; then release it.

The Application Editor displays the cell you moved with all branches intact as off-page connectors. (At this point, the status window is empty, which indicates that the cell is no longer selected.)

Figure 2-15 illustrates the HANG cell with off-page connectors after it is moved to page 2 of the drawing board. Figure 2-14 shows the HANG cell before and after it is moved.

Figure 2-15: Cell moved to another page



Note: You can select and move multiple cells at the same time. Refer to "Using the Edit Menu options" on page 2-32.

Removing a cell using the mouse

While you are laying out your design, you may find that you need to remove one or more cells from your application.

This section describes how to remove a cell by using the mouse. You can also use the Cut option on the Edit menu. Refer to the sections on using the Cut, Copy, Paste, and Move options starting on page 2-32.

To remove a cell using the mouse, follow these steps.

1 Select the cell by pointing and clicking MB1.

2 Choose the Delete option from the Edit pull-down menu (or press the keyboard accelerator <Ctrl+D>). The Application Editor removes the cell from the drawing board. In addition, this procedure removes all connections to the deleted cell.

3 Reconnect the surrounding cells.

Note: You can select and delete multiple cells at the same time by pressing the <Shift> key as you drag the mouse to select the desired cells.

Using the Edit menu options

Besides manipulating single cells in your application with the mouse, you can use the various Edit menu options to cut, copy, paste, or move an entire area of the drawing board. First, you must select the area to be edited. To select the area, follow these steps.

1. Create an anchor point by clicking MB1 and holding the <Shift> key at the point where you want the selection to begin.
2. Drag the mouse over the area you want to cut, copy, or move while holding down the <Shift> key and MB1.
3. The area you have selected will be indicated by a rectangle that appears on the screen.
4. Release the mouse button when the area you want to select appears within the rectangle.
5. This action ends the selection. Do not press MB1 again after you have ended the selection or you will deselect the selected area.
6. From the Edit menu, select the option you want to perform using the mouse or corresponding keyboard accelerator.

Using the Cut option

The Cut operation removes the selection area and deletes all elements within it. The accelerator keys <Ctrl+X> activate the Cut option. The selected cells are in the clipboard area.

Using the Copy option

The Copy operation copies the entire parameter structure of the cell in addition to the cell itself. When a group of cells is copied, only the connections between the group are retained; all other connections are lost. The accelerator keys <Ctrl+C> activate the Copy option.

Using the Paste option

The Paste operation pastes the entire area previously copied or cut to a new location. After you locate the new area for the selected cells, activate the Paste option. Move the pointer to the location where you want to paste your selection, and click MB2. To cancel a paste currently started, click MB3. The accelerator keys <Ctrl+V> activate the Paste option.

Using the Move option

Follow these steps. To reposition a selected area while retaining all the connections,

1. Using MB1, select the area you want to move. Note the rectangle that outlines your selection when you press MB1.
2. From the Edit menu, select Move. Alternatively, press the accelerator keys <Ctrl+M>.
3. Using MB1, move the rectangle that is displayed to the selected location, then release MB1.

   All connections are retained, and any necessary off-page connectors are automatically created.
4. To cancel a move that is currently enabled, click MB3.

Displaying text on the drawing board

You can add, remove, change, and move text on the drawing board. Select the “Add Text” option from the Edit menu.

Add Text option

The Add Text option adds a text block to the drawing board in Times Roman font in 8-10-12-14-18 or 24-point size. When you select the Add Text option, the Add Text dialog box appears as shown in Figure 2-16.

Adding text to the drawing board

To add text to the drawing board, follow these steps.

1. From the Edit menu, select the Add Text option.
2. Type the text in the top half of the dialog box.
3. Select the font type and size in the bottom half of the dialog box.
4. Click the Apply button.

5 Click MB1 to place the text at the location where you want the text to appear.

The text is displayed in a rectangle on the drawing board at the location you selected. You can move it by clicking MB2 on the text and dragging it to the new location.

You can activate the Add Text option by pressing the accelerator key sequence <Ctrl+t>.

6 To cancel an Add Text that is currently enabled, click MB3.

Change the Text option

The ChangeText option changes an area of text or a font already placed on the drawing board.

1 Double-click MB1 over the text you want to change.

The Text dialog box appears with the text you have chosen in the top half. You can modify the text and the font in the dialog box.

2 When you are finished, click the Apply button. The text appears on the drawing board in place of the text you selected. You can also select a new font.

3 To move text on the drawing board, select the text using MB2.

The text is replaced with a rectangle that contains the same dimensions.

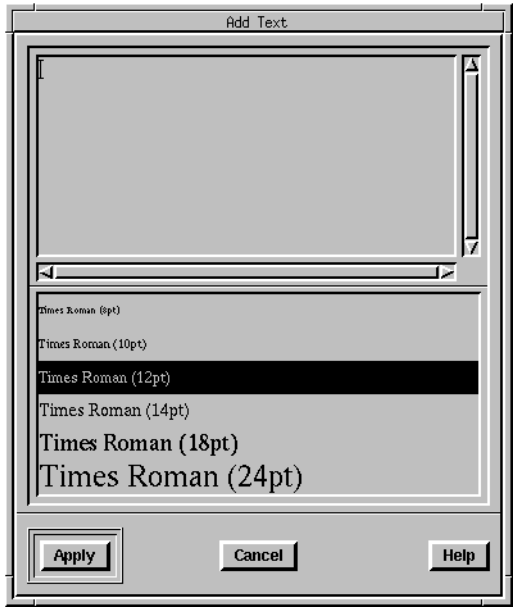
4 While pressing MB2, move the cursor to the location on the drawing board where you want the text to appear.

5 Release MB2. The text is displayed at the new location.

6 To remove text, press MB1 in the text region to select the text. The text is displayed in bold.

7 Press the backspace key to delete the text.

Figure 2-16: AddText dialog box



Insert and remove blank pages

If necessary, you can insert or remove pages in your application. These options are chosen from the Edit menu. The current visible page in your application is page 2.

Insert Page

The Insert Page option inserts a page at your current position in the application. You cannot insert pages before the first page because of the Start cell. To use the Insert Page option follow these steps.

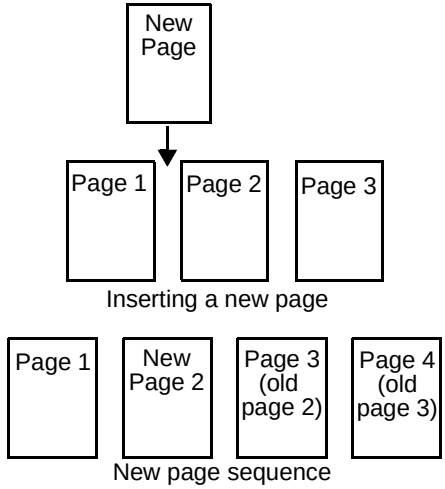
1 Choose the Insert Page option from the Edit menu.

2 A new blank page appears where page 2 was.

Page 2 and all pages occurring after page 2 and their contents move forward incrementally one page. Figure 2-17 shows how the Insert option works.

> ***Note:*** If you select the Insert Page option when the cursor is located on page 1, a page is not inserted.

Figure 2-17: How the Insert Page option works



Remove Page Option

The Remove Page option removes the current visible page in your application. There are two restrictions; first, page 1 of your application cannot be removed, and second, the page you are removing must be empty. The Remove Page option works in a similar way to the Insert Page Option: when you remove a page, all the pages following the removed page are incrementally moved back. The page that followed the removed page becomes the current page. Refer to the next section “Select All on Page Option,” for information about removing all text on a page.

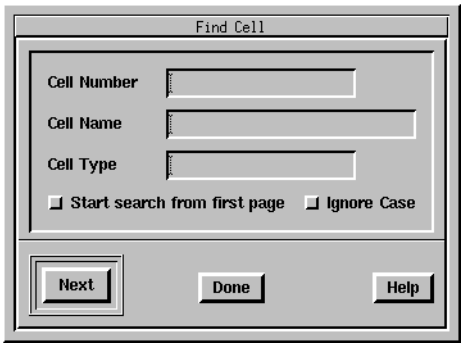
Select All on Page option

You can remove all the information on a particular page by choosing the Select All on Page option from the Edit menu and then pressing the <Delete> (backspace) key. Alternatively you can press the <Ctrl+A> accelerator keys to choose the Select All on Page option.

Finding a cell in your application

As your application grows in size, you may have difficulty remembering where a particular cell is located. You can search for a cell in your application by using the Find Cell option on the View menu. When you select the Find Cell option, the Find Cell dialog box appears as shown below.

Figure 2-18: Find Cell dialog box



You can search by cell number (which remains unchanged, even if you save and retrieve your application), cell name, or cell type.

Cell Number

Enter any valid cell number into this field.

Cell Name

Enter any valid cell name into this field.

Cell Type

Enter up to a four-character cell type into this field (for example: ANSW, DOUT).

> ***Note:*** If you delete cells, the associated cell numbers are no longer available for your application.

The three push buttons at the bottom of the dialog box have the following functions:

Next. Finds the next occurrence of your entry. You can press this button until the last matching occurrence is found.

Done. Closes the dialog box.

Help. Gives detailed help on the Find option.

Saving and naming your application

As a matter of prudent design, you should save the application on the drawing board often. You have the option of saving the application under its current name (the name displayed in the "Current Application" box in the Control Area) or under a new name.

Saving an application under the current name

To save the application under its current name, follow these steps.

1 Point and click MB1 on the File option on the menu bar.

The Application Editor displays the File pull-down menu.

2 Point and click MB1 on the Save option.

The Application Editor saves the application.

> ***Note:*** (As an alternative to Steps 1 and 2, you can type the keyboard accelerator <Ctrl+S>.)

If this is a new application and you use the Save option (instead of the Save As option) to save the application, the Application Editor saves it under the current name, "Untitled."

Saving an application under a new name

To save the application under a new name, follow these steps.

1 Click MB1 on the File option on the menu bar.

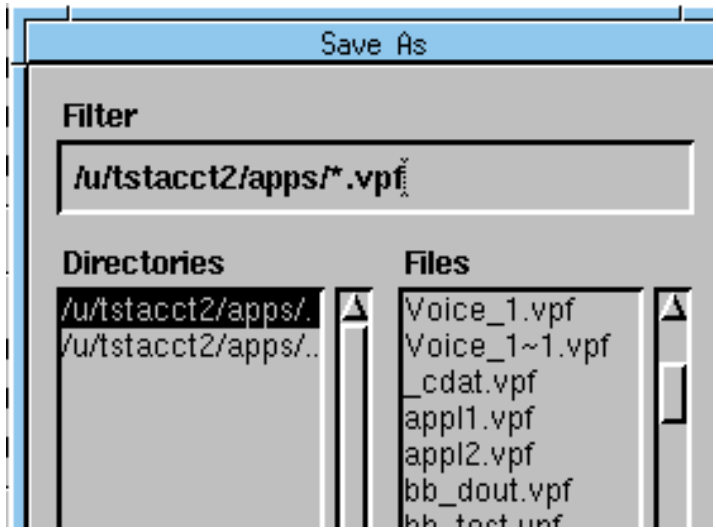
The Application Editor displays the File pull-down menu.

Note: (As an alternative to Steps 1 and 2, you can use the keyboard accelerator <Ctrl+A> to display the File Browser.

2 Click MB1 on the Save As option.

3 The Application Editor displays the File Browser.

4 Enter a new name in the text entry box as shown below and point and click MB1 on the OK button.



The Application Editor saves the application on the drawing board to the new file name.

Note: For further information on using the Browser, refer to “Creating and selecting an application file”.

5 To save a new application, use the Save As option in the File menu.



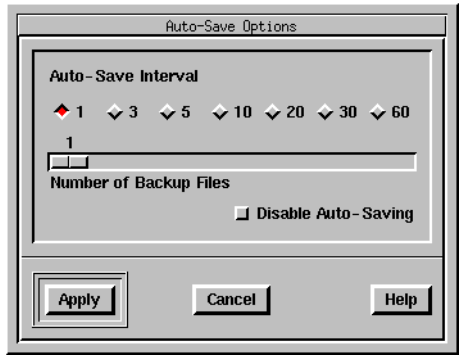
CAUTION
Risk of data loss

By choosing the Save As option and specifying an application name, you avoid inadvertently saving the application as "`Untitled.vpf`"—a name that does not describe the application and that can be easily overwritten when you later save another new application.

Saving your application automatically

You can direct Application Editor to automatically save your application at specified intervals while you edit it. To do this, select the Auto-Save option from the Preferences menu. This displays the Auto-Save Options window.

Figure 2-19: Auto-Save Options Window



Backup files reside in the apps directory. The Application Editor will save each backup file with the same name you have given your application, designating backup copies by appending the name, which may be truncated to seven characters, with a tilde (~) and the number of the backup. The extension is always .vpf.

For example, if you have named your application `outcalling`, the Application Editor will call the first three automatically saved copies:

- `outcall~1.vpf`
- `outcall~2.vpf`
- `outcall~3.vpf`

The Auto-Save option presents Interval buttons that offer choices (in minutes) of 1, 3, 5, 10, 20, 30, and 60, as shown in Figure 2-19. Click your selection.

There is also a slide bar with which you can select the number of backup files you want to store. You can designate up to 10.

You can disable this feature by clicking a button in the same option menu.

Printing your application

The Application Editor provides a facility for creating a graphical Postscript representation of your application along with a listing of cell parameters, databases, user functions, prompts, and buffers used by the application. Each page of the drawing board prints as a single 8.5 x 11 page.

To print your application, follow these steps.

1 From the File menu, choose the option “Export as PostScript”.

2 The following options are displayed: All, Call Flow, and Cell Contents.

3 Select one of the options from the submenu:

Option	Description
All	Creates a PostScript file consisting of the call flow and the cell contents.
Call Flow	Creates a graphical PostScript representation of your call flow only.
Cell Contents	Creates a PostScript file containing the parameters for every cell in your call flow.

Note: As an alternative to steps 1 and 2, you can type the keyboard accelerator <Control+P>, which has the same effect as selecting All from the submenu.

Note: The Application Editor generates a PostScript file (in the apps directory of AP software) with the same name as your application with the `.ps` extension.

4 Move the pointer to another window running the shell.

5 Enter the appropriate print command for printing `.ps` files (site-specific) at the command prompt.

Note: You must upload this file to a PostScript printer.

Executing your application

Once you save your application, it can be loaded on a port and run immediately. Compilation is not required.

Note: Any application you build with the Application Editor will execute, no matter how erroneous the logic is. Consequently, you should test your application to ensure that your objectives are achieved during run time.

Debugging an application

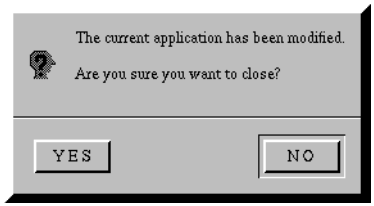
AP software and the Application Editor include the OPEN IVR application Graphical Debugger. You can use this graphical tool to identify and modify problems within a selected application, channel, or combination of an application with a particular channel. Refer to Chapter 5, “Building a sample application,” for additional information about using the Graphical Debugger.

Closing an application

If you want to “clear” the drawing board without exiting the Application Editor, close the current application, by following these steps.

1 From the menu bar, select File>Close. As an alternative, you can type the keyboard accelerator <Ctrl+L>.

The Application Editor clears the drawing board, displaying its initial set up with only the START cell, unless you have made changes since last saving the application. If you have made changes since you last saved the application, the Application Editor prompts you with the following window: .



2 To close without saving changes, click YES. To cancel the close operation, click NO.

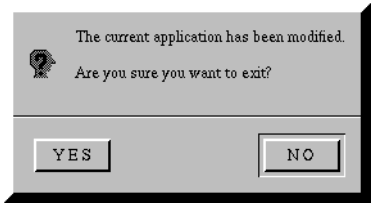
Exiting the Application Editor

To exit from the Application Editor, follow these steps.

1 From the menu bar, select File>Exit. As an alternative, you can type the keyboard accelerator <Control+E>.

The Application Editor does not perform a Save/Exit operation. If you have made changes to the application since the last save, the Application Editor displays the dialog box illustrated in Figure 2-20. Select YES to exit the Application Editor without saving the changes, or NO to cancel the exit operation and return to the drawing board.

Figure 2-20: Exit without saving changes



Note: The toggle button for the Save setting on Exit selection under the Preferences menu allows you to set your applications settings to save automatically when you exit the Application Editor. The next time you open the application, the same preferences are used. Refer to the discussion of options in the “Preferences menu” section on page 2-23.

Application development hints

To help you develop a strategy for creating your application, consider the following:

- The call flow enters the cell through the input branch in the top of the cell.
- Input branches direct the call flow into the cell only. Exception and Success branches direct the call flow out of the cell only.
- You do not have to connect cells in the order you placed them.

- While the branch connect tool is active (and the pencil cursor is displayed, therefore also referred to as pencil mode), you can select multiple cell connectors.

  However, the branch is completed and pencil mode is deactivated when two sequentially selected cell connectors correspond to a sourcedestination pair. What actually happens depends on whether or not the `Record Branches` option is on (refer to the Preferences menu). When `Record Branches` is off, a single branch is formed. When the option is on, branches between multiple output connectors and a single input connector can be formed.

  For example, if the `Record Branches` option is off and you turn on pencil mode, selecting an output connector on cell 3, an output connector on cell 5, and then an input connector on cell 8 will create a single branch between cells 5 and 8. However, if the `Record Branches` option is on and you turn on pencil mode, selecting an output connector on cell 3, an output connector on cell 5, and then an input connector on cell 8 will branch cells 3 and 5 to cell 8.

- A branch that exits a cell from one of the sides of the cell icon is an Exception branch and is displayed as a dashed (red) line. A branch exiting from the bottom of the cell indicates success and is displayed as a solid (green) line.

- When you are creating an Exception branch, you can select either side of the cell. However, the Application Editor automatically selects the side that will result in the shortest connection.

- Each source connector must have a destination; otherwise, when you save your application, the Application Editor will warn you about disconnected cells.

- Pencil mode remains active across pages in your application. This feature is useful when you want to connect cells on different pages. Whether you are working with cells on the same page or on different pages, pencil mode is active until you either turn it off or complete a source-destination pair (refer to “Off-page connectors” on page 2-27).

  Note: While pencil mode is active, pressing MB1 on a connector highlights the connector in red until you release the mouse button.

Chapter 3: All about cells

OPEN IVR applications are made up of cells. A cell is a graphical representation of a small piece of your application's functionality. You create OPEN IVR applications by connecting cells.

All AP software cells share a similar basic structure but have functions ranging from answering a telephone call to sending a fax to accessing information on a remote computer. AP software cells are organized into groups of similar functionality.

This chapter gives you an overview of all the AP software cells and describes their functions.

Components of a cell

Each AP software cell has two main components:

- Cell icon
- Cell Parameters page

Cell icon

The cell icon is the graphical representation of the cell that you place on the drawing board. The cell icon displays the:

- Cell group
- Cell name
- User-defined name
- Cell number
- Branches

Cell group

AP software cells are organized in groups of similar functionality. A graphic on the cell palette represents this functionality. This graphic also appears in the upper left of each cell icon. This graphic informs you of the basic type of function this cell performs.

Cell name

The cell name is the specific title of the cell. The title consists of a three to five-letter code as well as a phrase that describes the cell's specific function. This title appears at the top of the cell icon.

User-defined name

The cell icon contains a field in which you can enter a word or phrase to provide application specific information about that particular cell. To enter text in this field, you must access the cell Parameter Page (explained later).

Cell number

The cell number provides a unique identification number for each cell in your application. When you place a cell on the drawing board, the cell is automatically numbered by AP software. This number remains on this cell for the life of the application. Numbers for deleted cells are no longer referenced in the application.

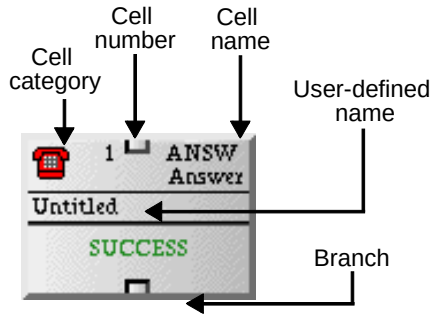
Branches

Each cell in an OPEN IVR application connects to other cells through branches. Each branch has a name describing the event that sends the call flow to that branch. For example, most cells have a success branch. If AP software successfully executes that cell, the call flow takes the success branch. The success branch should lead to the next cell in the call flow.

Each cell has a branch for every event that can occur during its execution. Cells that request caller input have branches for time-out, invalid input, and errors. Cells that perform mathematical comparison of buffer values have branches for greater than, less than, and equal to. The types of branches depend on the cell's function.

Figure 3-1 shows a basic cell icon and its component parts.

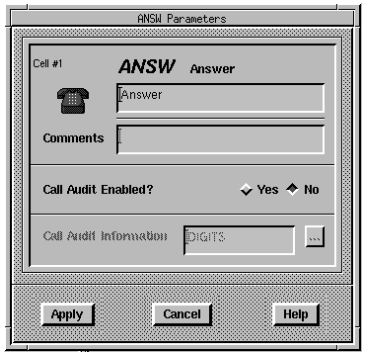
Figure 3-1: Components of a cell icon



Parameters page

Each cell has a Parameters page that you use to tune the cell for your application. The Parameters page gives you control over how the cell performs its function. You access the cell page by double-clicking mouse button 1 on the cell icon. Figure 3-2 shows the Parameters page for the ANSW cell.

Figure 3-2: Answer cell parameters page



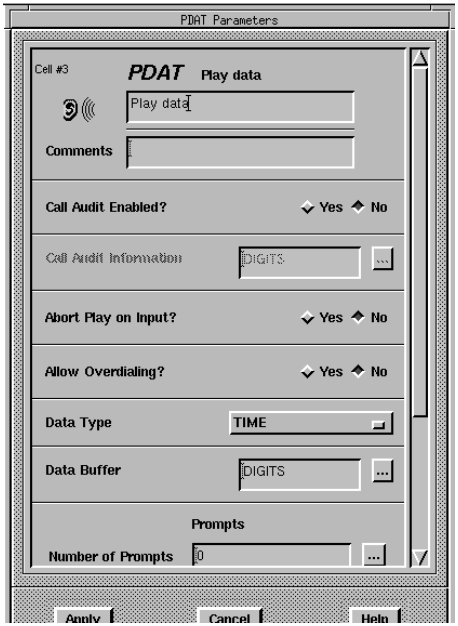
Examine the radio buttons and the text entry field in Figure 3-2. When the Radio Button in the No position, the text entry field below it is grayed out. When the radio button is in the Yes position, the text entry field is active. If you change the entry in the text box while the field is active and then deactivate the field with the radio button, the value in the field is returned to its default value.

The upper half of the ANSW cell Parameters page is a header displaying the cell number, cell name, cell category, and two text entry fields. In the upper text entry field, enter the user-defined name. The entry in this field will appear on the cell icon. If you do not enter anything in this field, the word “Untitled” remains on the cell icon. In the Comments field, you can enter a comment if you choose. This comment appears only on the Parameters page.

The lower half of the ANSW cell Parameters page contains the parameters governing Call Process Auditing. You manipulate these parameters with radio buttons and a text entry box with a browser. Refer to the *Symposium OPEN IVR System Administration Guide* Release 4.0, for information about the Call Auditing parameter.

Since all the ANSW cell does is answer a call and start the call flow, it has very few user-definable parameters. Figure 3-3 shows the Parameters page for the Play Data (PDAT) cell, which contains more parameters that you can tune.

Figure 3-3: Play Data cell parameters page



Note that the Play Data cell Parameters page shares header layout and Call Process Auditing parameters with the Answer cell Parameters page. All cells share this header layout and Call Process Auditing parameters (except for the EVENT, cell which does not have Call Process Auditing parameters. See the *Symposium OPEN IVR Cell Catalog* Release 4.0, for more information on the EVENT cell).

START cell

The START cell is displayed on the cell palette by the Application Editor. When you double-click MB1 in the START cell, the DFLT cell (Default) is displayed. You can set global preferences for your application in the DFLT cell. For example, you can do the following:

- Select a message database for your application.
- Establish global parameter values for your application.
- Allow or disallow global functions in your application.
- Speed up and slow down the transcription of audio information into text.

Refer to “Establishing application defaults” on page 3-11 to see a complete list of options in the DFLT parameters cell.

Identify an SQL DBMS and database for the application to access, as well as the node name of its location on the network.

> ***Note:*** If you specify an SQL database, specify a valid user name and password to enable your application to access the DBMS.

Modifying DFLT cell parameters

To display and modify the DFLT cell parameters, follow these steps.

1 Double-click MB1 on the START cell.

2 Or, if you are on another page of the drawing board, follow these steps:

a Select the View option on the Application Editor window menu bar.

b Select the option Default Cell.... Note that the START cell is also referred to as the DFLT (Default) cell.

The Parameters page for the START cell is displayed.

3 Many parameter values are displayed in text entry boxes. For these parameters, you can modify the value directly by selecting the text entry box and typing a new value in the box, or by pointing and clicking MB1 on the associated ellipsis button to select a value from the Browser.

4 In order to reassign the default value <NONE> in the absence of a browser, type the value **<NONE>** in the text field.

5 Some parameters are displayed as digit buttons. When you point and click MB1 on a digit button, the Application Editor displays the touch-tone keypad window illustrated in Figure 3-7.

6 You can click MB1 on any digit to designate a new default value, including None to indicate that there is no default digit.

Once you select a digit, the touch-tone pad window disappears and the original digit button is shown with the new value.

7 The parameter Get Data Termination is a text field. If you click MB1 on the ellipsis button, then the system displays the touch-tone keypad window illustrated in Figure 3-7. You can click MB1 on the digits *, #, or A-D to either select or deselect those digits. You cannot select digits 0-9. Deselected digits are blank, selected digits are highlighted in black and nonselectable digits are grayed out.

8 Once you select the Get Data Termination digits, click the Apply button to save this information and update the text field. Clicking None or Apply without selecting a digit(s) results in the default as no digits. Clicking the Cancel button cancels any changes made to the parameter Get Data Termination digits.

Figures 3-4, 3-5, and 3-6 show the DFLT cell parameters.

Figure 3-4: START cell parameters (part 1)

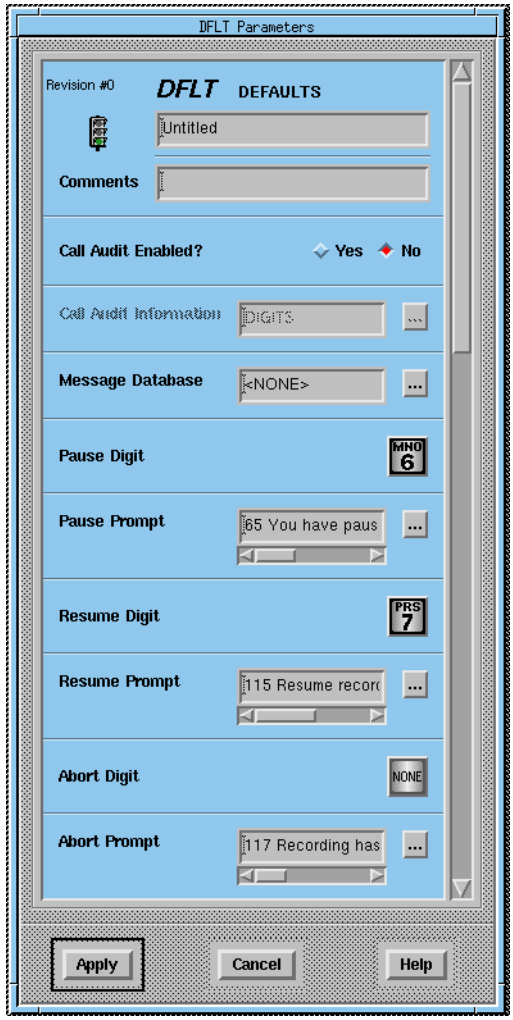


Figure 3-5: START cell parameters page (part 2)



Figure 3-6: START cell parameters (part 3)

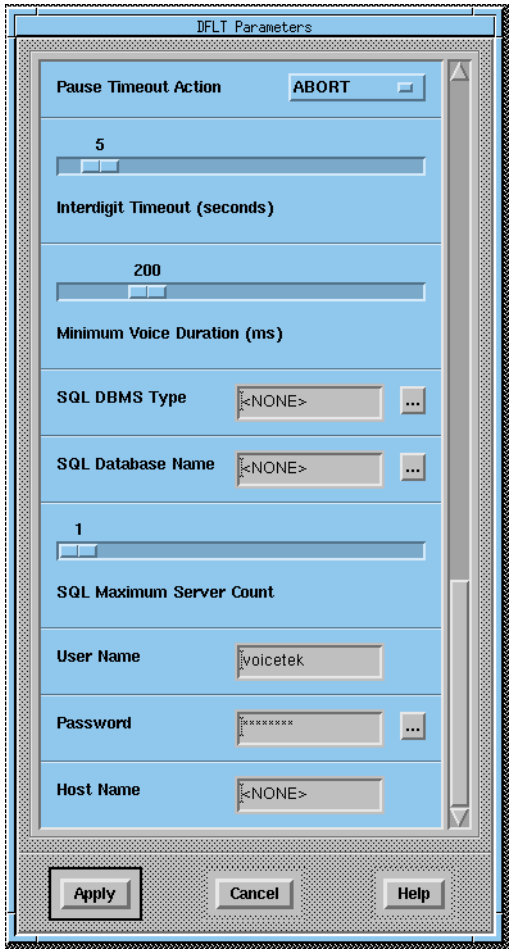


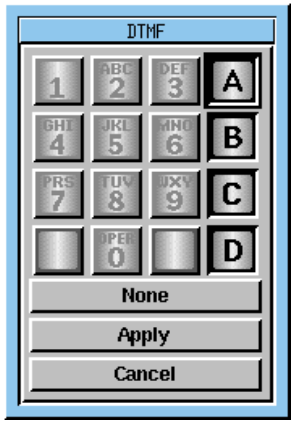
Figure 3-7 illustrates the Default cell `Get Data Termination` touch-tone keypad window.

Figure 3-7: Default cell Get Data Termination touch-tone keypad window



Select only A, B, C, or D if your callers' telephones support these keys. Figure 3-8 displays the Default cell Get Data Termination touch-tone keypad window with the keys A, B, C, and D selected as termination digits.

Figure 3-8: Default cell Get Data termination touch-tone keypad window



Note: The touch-tone pad associated with the Parameters page is unique. Other touch-tone pad windows displayed by the IVR Developer (for example, when you display the parameters for other cells) show a “Default” key in place of the “None” key. In this case, selecting the “Default” digit displays the value Default in the text field, indicating that the parameter uses the value defined by the application default.

The values for other parameters in the Application Defaults window are indicated by a slider. You can change the current value by pointing and dragging mouse button 1 to a different value.

Establishing application defaults

The following items appear on the START Cell Parameter Page:

Audit Enabled	Audit Information
Message Database	Pause Digit
Pause Prompt	Resume Digit
Resume Prompt	Abort Digit
Abort Prompt	Termination Digit
Skip Ahead Digit	Back Track Digit
Skip To End Digit	Back To Beginning Digit
Speedup Digit	Slowdown Digit
Get Data Termination	Maximum Pause Time
Skip Ahead/Back Track Time	Pause Timeout Action
Interdigit Timeout	Minimum Voice Duration
SQL DBMS Type	SQL Database Name
SQL Maximum Server Count	User Name
Password	Host Name

Example

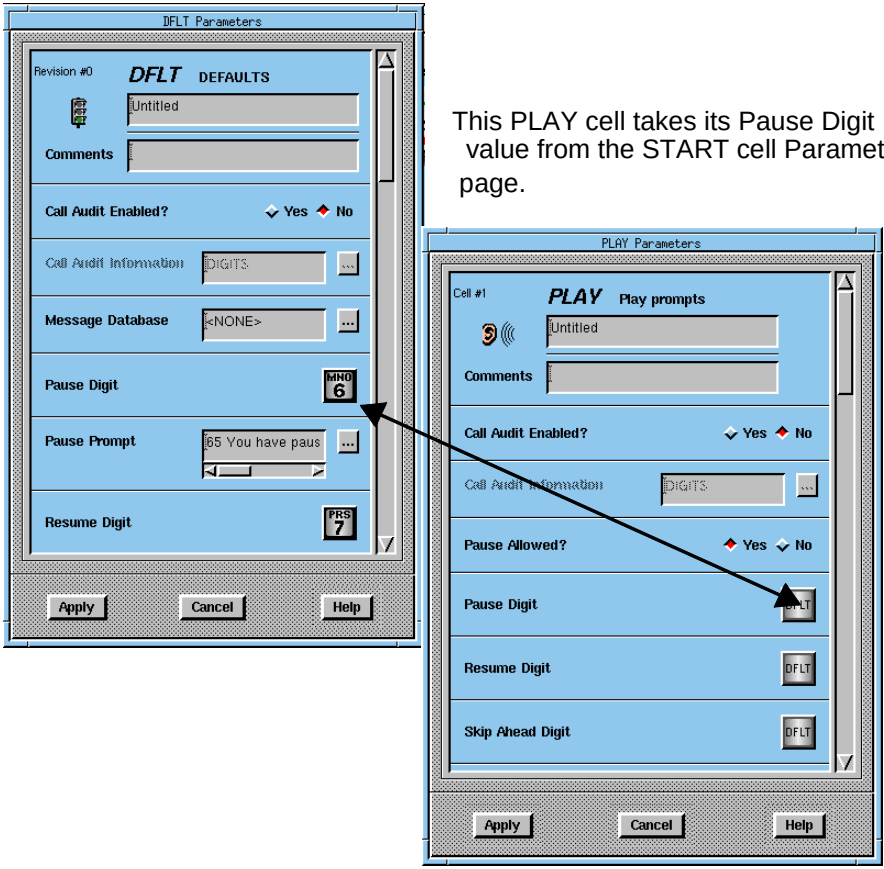
The following example explains how to use all of the parameters in the default cell.

An application uses a PLAY cell to play prompts to a caller. One of the parameters in the PLAY cell is `Pause Digit`. The value you specify for this parameter determines the key that the caller presses to pause the application temporarily while playing the prompts. You can enter any DTMF key from 0 to 9, *, or # as the `Pause Digit` parameter. Instead of entering a DTMF key, you can also select DEFAULT as the value of the `Pause Digit` parameter. If you select DEFAULT, then the value of Pause Digit in the cell is taken from the `Pause Digit` in the DFLT Parameters, as shown in Figure 3-9. The PLAY cell in Figure 3-9 is assigned DEFAULT in the Pause Digit field. Since a value of 6 is assigned to the parameter Pause Digit in the DFLT cell, the PLAY Cell uses 6 as its Pause Digit.

If your application uses several PLAY cells, you can set the value of `Pause Digit` in the START cell that applies to all the PLAY cells simultaneously. If you change the value on the START cell Parameters page, you automatically change the value of the Pause Digit in any PLAY cell where the Pause Digit is assigned a value of DEFAULT. Figure 3-9 shows the PLAY cell taking its value from the Default cell window.

Note that the parameters `Pause Digit` and `Resume Digit` parameters are also valid for the following cells: PMSG, RECP, and RMSG.

Figure 3-9: Default values and the Default cell



The use of default values wherever possible ensures that key actions and timeout values remain the same throughout your application.

Application default fields

The following sections explain each default field in the START cell.

Message database

Default value	<NONE>

This parameter indicates the name of the database in which AP software stores messages left by callers. Only the names of the messaging databases that are defined in the applications are displayed when you click "**...**" in the Message Database parameter. To see the names of the message databases, access the OPEN IVR Information Database Editor from the Control Panel.

> ***Note:*** Enter the name of the database without brackets (that is, enter the database name in the format *database_name*; do not enclose the database name in brackets).

Pause digit

Default value	6
Valid values	NONE, any DTMF key

You can create an application that allows a caller to record or play back a prompt or a message. If you do so, you may want the caller to be able to halt the recording or playback process temporarily by pressing one of the telephone keys. The caller must press the pause digit to perform any of these actions. The pause digit is an option for the following cell types:

- PLAY (Play Prompts)
- PMSG (Play a Message)
- RECP (Record Prompt)
- RMSG (Record Message)

> ***Note:*** When choosing the pause digit, it is recommended that you use the 5 and 7 keys with caution because they are often falsed; that is, certain frequencies commonly found in speech recordings may be incorrectly interpreted as the DTMF digits 5 and 7.

The initial value of the pause digit is 6, but the digit can be any DTMF key or NONE.

Pause prompt

Default value	65
Prompt description	You have paused recording. To resume, press...
Valid values	DISABLE, a prompt number

You can enter the number of a prompt, if any, that the caller hears after pressing the pause digit. The pause prompt is an option for the following cell types:

- RECP (Record Prompt)
- RMSG (Record Message)
- PLAY (Play Prompts)
- PMSG (Play a Message)

The initial value is 65; this is a standard prompt saying, “You have paused recording. To resume, press...”. Any prompt number is valid. You can also use DISABLE as the value of the Pause Prompt field if you do not want any prompt to be played when the caller presses the Pause Digit.

Resume digit

Default value	7 (see note below)
Valid values	NONE, any DTMF key

If you have a pause digit, you must choose the resume digit the caller presses to resume recording or playback. Like the pause digit, the resume digit is an option for the following cell types:

- PLAY (Play Prompts)
- PMSG (Play a Message)
- RECP (Record Prompt)
- RMSG (Record Message)

Note: When you choose the resume digit, use the 5 and 7 keys with caution because they are often falsed; certain frequencies commonly found in speech recordings may be incorrectly interpreted as the DTMF digits 5 and 7.

If you specify the value `NONE`, then the "Resume Prompt" parameter is disabled.

The initial value of the resume digit is 7, but the digit can be any DTMF key.

Resume prompt

Default value	115
Valid values	DISABLE, a prompt number

Note: If you specify `NONE` as the value in the "Resume Digit" parameter, then the "Resume Prompt" parameter is disabled.

You can specify the number of the prompt (if any) that the caller hears after pressing the resume digit. The resume prompt is an option for the following cell types:

- RECP (Record Prompt)
- RMSG (Record Message)
- PLAY (Play Prompts)
- PMSG (Play Message)

The initial value is set to 115, which is the number for the standard prompt saying: "Resume recording at the tone." Any prompt number is valid.

Abort digit

Default value	NONE
Valid values	NONE, any DTMF key

The abort digit, like the termination digit, is a key the caller can press to stop recording a prompt or a message. The abort digit does not save the recording, and it cannot be added to the message database; the message is deleted. Like the termination digit, the abort digit is an option for the following cell types:

- RECP (Record Prompt)
- RMSG (Record Message)

Note: When you choose the abort digit, you should use the 5 and 7 keys with caution because they are often falsed; that is, certain frequencies commonly found in speech recordings may be incorrectly interpreted as the DTMF digits 5 and 7.

The initial value is `NONE`, but the digit can be any DTMF key or `NONE`. If you accept the default value `NONE`, then the "Abort Prompt" parameter is disabled.

Abort prompt

Default value	117
Prompt description	Recording has been aborted. The message will not be saved.
Valid values	DISABLE, any prompt number

Note: If you accept the default value `NONE` in the "Abort Digit" parameter, then the "Abort Prompt" parameter is disabled. The prompt number is displayed, but it is not accessed.

This is the number of the prompt (if any) that the caller will hear if playback or recording is aborted. As mentioned in preceding paragraphs, this may occur if the caller has exceeded the Maximum Paused Time and you have selected ABORT as the Pause Timeout Action. The abort prompt is an option for the following cell types:

- RECP (Record Prompt)
- RMSG (Record Message)

Termination digit

Default value	#
Valid values	NONE, any DTMF key

The termination digit is the telephone key the caller can press to finish recording a prompt or a message. When used to end recording, the recording is saved and can be added to the message database. The termination digit is an option for the following cell types:

- RECP (Record Prompt)
- RMSG (Record Message)

Note: When you choose the termination digit, you should use the 5 and 7 keys with caution because they are often falsed; that is, certain frequencies commonly found in speech recordings may be incorrectly interpreted as the DTMF digits 5 and 7.

The initial value of the termination digit is #, but the digit can be any value from 0 to 9, *, #, or NONE.

Skip ahead digit

Default value	NONE
Valid values	NONE, any DTMF key

The skip ahead digit allows the caller to specify a digit that, when pressed while listening to a message or a prompt during a call, will move forward within the message or prompt a certain amount of time. The caller can then continue to listen to the prompt/message from the advanced point. The amount of time that the prompt/message should be skipped ahead is specified in the field Skip Ahead/Back Track Time. The skip ahead digit is an option for the following cell types:

- PMSG (Play a Message)
- PLAY (Play Prompts)

Note: When choosing the skip ahead digit, it is recommended that you use the 5 and 7 keys with caution because they are often falsed; that is, certain frequencies commonly found in speech recordings may be incorrectly interpreted as the DTMF digits 5 and 7.

Backtrack digit

Default value	NONE
Valid values	NONE, any DTMF key

The back track digit allows the caller to specify a digit that when pressed while listening to a message or a prompt during a call, will rewind the message or prompt a certain amount of time. The caller can continue to listen to the prompt/message from that point on. The amount of time that the prompt/message should be rewound is specified in the field Skip Ahead/Back Track Time. The back track digit is an option for the following cell types:

- PMSG (Play a Message)
- PLAY (Play Prompts)

Skip to end digit

Default value	NONE
Valid values	NONE, any DTMF key

The skip to end digit allows the caller to specify a digit that when pressed while listening to a message or a prompt during a call, will skip and go to end of the message/prompt. The caller can continue to listen from that point on if there are more messages/ prompts. The Skip To End digit is an option for the following cell types:

- PMSG (Play a Message)
- PLAY (Play Prompts)

Back to beginning digit

Default value	NONE
Valid values	NONE, any DTMF key

The back to beginning digit allows the caller to specify a digit that when pressed while listening to a message or a prompt during a call, will rewind and go to the beginning of the message/prompt. The caller can listen to the message/prompt again using this feature. The Back To Beginning digit is an option for the following cell types:

- PMSG (Play a Message)
- PLAY (Play Prompts)

Speedup digit

Default value	NONE
Valid values	NONE or any DTMF key

The speedup digit enables the caller to specify a digit that when pressed while listening to a message or a prompt during a call, will fast-forward the message/prompt. The caller can press this digit several times while he or she is listening to a message/prompt until the end of the message/prompt is reached. The speedup digit is an optional parameter for the following cells:

- PMSG (Play a Message)
- PLAY (Play Prompts)

Slowdown digit

Default value	NONE
Valid values	NONE, any DTMF key

The slowdown digit enables the caller to specify a digit that when pressed while listening to a message or a prompt during a call, will slow down playing the message/prompt. The caller can do this several times while he or she is listening to a message/prompt till the end of the message/prompt is reached. The slowdown digit is an option for the following cell types:

- PMSG (Play a Message)
- PLAY (Play Prompts).

Get Data Termination

Default value	* #
Valid values	None, or any combination of * A B C D

The Get Data Termination digit is the telephone key the caller can press to finish inputting data. The `Get Data Termination` parameter applies to the GDAT (Get Data) cell.

Maximum Pause Time

Default value	15
Valid values	1 to 75

If the caller uses the pause digit, then recording or playback is paused temporarily. Specify the maximum length of time in seconds that the caller can pause. If this time is exceeded and the caller has not pressed the resume digit, then the pause timeout action takes effect. The `Maximum Paused Time` parameter applies to the following cell types:

- PLAY (Play Prompts)
- PMSG (Play a Message)
- RECP (Record Prompt)

- RMSG (Record Message)

Skip ahead/backtrack time

Default Value	2
Valid Values	2 to 30 seconds

If the caller uses the skip ahead or back track digit, then playback of messages/prompts will be skipped ahead/back tracked ahead the amount of time specified in this field. For example, if the value specified here is 4 seconds, then when the caller presses the Skip Ahead Key, the message/prompt will be skipped ahead 4 seconds and the caller can continue to listen to the rest of the message. The Skip Ahead/Back Track Time parameter applies to the following cell types:

- PLAY (Play Prompts)
- PMSG (Play a Message)

The initial value is 2, but the time can be any value from 2 to 30 seconds.

Pause timeout action

Default value	ABORT
Valid values	ABORT, TERMINATE

A caller may use a pause digit to pause or play back a recording of a prompt, there is a limit on the pause time. When you build an application, it is up to you to decide what the application should do if a caller pauses during the recording process and then exceeds the maximum pause time. The choices are abort or terminate.

Abort stops the recording; the recording is not saved and it cannot be added to the message database.

Terminate stops the recording; the recording is saved and can be added to the message database.

A pause timeout action must be set for the following cell types if the pause option is used:

- RECP (Record Prompt)

- RMSG (Record Message)

The initial value is ABORT; TERMINATE is also valid.

Interdigit timeout

Default value	5
Valid values	0 to 75 seconds

A value of 0 means there is no time limit.

An application may require a caller to enter information by pressing telephone keys to produce DTMF digits. If so, you must specify the maximum amount of time that can elapse between the end of the prompt and the caller pressing the first DTMF digit. By setting a time limit, you can control any problems that might arise if a caller does not enter digits, cannot enter digits, or fails to enter as many digits as requested.

The interdigit timeout value affects the following cell types:

- GDAT (Play Prompts and Get Data)
- MENU (Play Menu and Get Data)

The initial timeout value is 5, but the timeout value can be any value from 0 to 75 seconds, with a value of 0 meaning there is no time limit.

Minimum Voice Duration

Default value	200 milliseconds
Valid values	50 through 750 in 50-millisecond increments

The Default Cell screen has an item called Minimum Voice Duration. This parameter is used to enable speech confirmation, an option for MENU cells. The Minimum Voice Duration parameter in the Default Cell sets the minimum valid duration of speech that can be accepted for positive confirmation. For more details, see the description of the MENU cell in the Cell Catalog. The value of this parameter affects all MENU cells that have the Voice Confirmation feature enabled.

The initial value is 200 milliseconds. The parameter's value can range from 50 to 750 in increments of 50.

SQL DBMS type

Default value	<NONE>

Note: The SQL fields in the Default cell Parameters page work together to specify a set of Generic Data Server (GDS) processes for your AP software applications to communicate with. Refer to the *Symposium OPEN IVR SQL Server User Guide* Release 4.0, for information about configuring these parameters.

SQL database name

Default value	<NONE>
Options	System dependent

Note: Refer to the *Symposium OPEN IVR SQL Server User Guide* Release 4.0, for additional information about supported SQL databases. If you specify an SQL database name, ensure that you specify valid entries in the parameters `User Name` and `Password` in order to enable your application to access the specified DBMS.

SQL maximum server count

Default value	1
Valid values	1-99

This parameter sets the number of SQL sessions that are connected to the server you choose. During application development, use the default value of 1. However, when you prepare the application for production, you need to analyze performance versus load input, especially as it pertains to database concurrency, when you specify a value for this parameter. Although you can specify values up to 99, use caution when you enter a value. As the value is increased, the load on the system and the concurrency of the database are also increased.

Note: System performance is significantly affected by increasing the number of GDS processes. Increase the number of GDS processes only when necessary.

User Name

Default value	nortel

This field should contain a user name with access to the SQL database. Refer to your third-party database package for information on defining user names and passwords. Without a valid user name, your application cannot access the SQL database.

Password

Default value	nortel

This field contains an encrypted password that AP software uses to access the SQL database. The password in this field appears as asterisks so that only the person who entered the password knows what it is. To change the password, enter it in this field. The characters you type appear as asterisks. To verify the password you entered, click the button to access the browser. Instead of getting a browser, you will get a password verification pop-up window. Enter the password again in this pop-up (again the password will be encrypted with asterisks). The passwords must match for AP software to implement the new password. Without a valid password, your application cannot access the SQL database.

Host name

Default value	<NONE>

The host name is the node name of the system that contains the SQL database that resides on the network.

Note: You must include the IP address for this node in the file `/etc/hosts`.

This field contains the system identifier for a remote database to which you want to connect. On a stand-alone system, use the default value (NONE).

Applying the new default parameters

Once you select the default values for your application, you can save them and return to the drawing board by pointing and clicking MB1 on the Apply push button. If you redisplay the default parameter values, you will see the new values (also, any values that you typed in the entry boxes will be included in the Browser list of valid values). You can change the default values at any time while designing your application. If needed, you can also change the default values within each cell that utilizes default values.

Establishing the cleanup handler cell

There are activities that may need to be performed after the phone call ends. The cleanup handler is a subapplication designed to be executed after a HANG (Hang Up) cell ends the call or after the caller ends the call by hanging up.

A cleanup handler consists of one or more cells. It may perform activities on the application server, such as updating databases or compiling statistics about the phone call. The channel on the MRS remains in an off-hook state until the cleanup is completed. Then the HANG cell is accessed; the cell executes a command to release the channel on the MRS. Note that a cleanup handler performs activities on the AP only.

Note: Keep the following information in mind when you create a cleanup procedure:

- The cleanup handler cannot communicate with the MRS except through the DMSG cell. Use of telephony cells except DMSG in the cleanup branch should be avoided.
- The last cell in the subapplication must be a HANG cell with 0 prompts played.
- The following cells must not be placed in the cleanup handler: RETN, GSUB, or EXEC.
- Lengthy cleanup procedures, which can cause the telephone company to maintenance-busy the trunks, should be avoided.

Chapter 4: Concepts of application design

This chapter gives you basic design strategies for creating successful voice applications as well as information on buffers and OPEN IVR's built-in database system.

When you build an application, you create a call flow by placing cells on the drawing board, connecting them, and customizing each cell to suit your application's needs. When you connect a cell, you connect its different branches. A branch is a pathway from one cell to another. Since a cell may have several outcomes, there may be several branches out of a cell. Buffers temporarily store information your application may need so that it can be accessed by any subsequent cell. Two kinds of built-in databases store either voice messages, faxes, or information. OPEN IVR can access third-party databases by using SQL relational database queues. Prompt numbers and message numbers identify voice recordings.

Application components

OPEN IVR uses the following components as building blocks for applications:

- cells
- branches
- database access
- prompt numbers and message numbers
- interface to user-defined code

An application may be run on one or many telephone ports simultaneously. The application normally begins by answering or making a telephone call and ends by hanging up or transferring the call. When you build an application, you decide how your application begins, handles, and concludes.

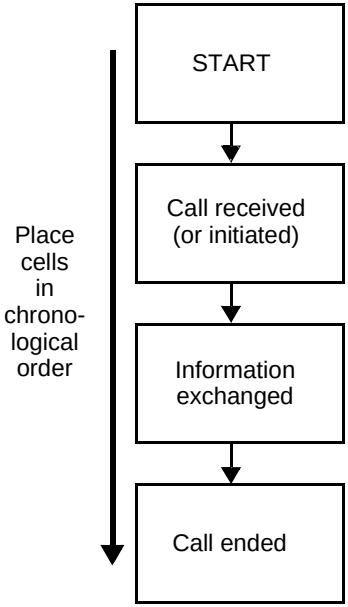
Application basics

Begin your application by connecting the first cell in your application to the output branch of the START cell. Then place and connect the rest of your call flow. After you save the application, it is ready to be loaded and run on one or more telephone ports.

Normal call flow is top to bottom, as shown in Figure 4-1. Once you select cells from the palette and place them on the drawing board, you can move them anywhere and interconnect them by drawing lines between the branch connectors. Cells may be entered only at the top at the entry connector, whereas branches can be designated from the bottom (success) or sides (exception).

Whether you connect an exception branch from the left or the right side of the cell, the Application Editor automatically selects the branch connector that minimizes the length of the connection between the two cells.

Figure 4-1: Top-to-bottom call flow



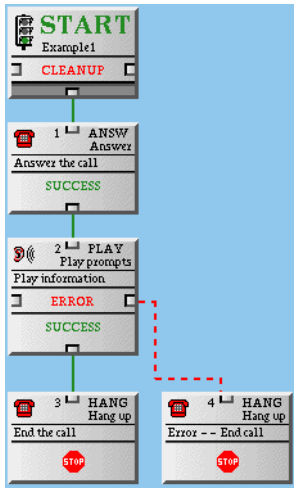
Hang cells

In long, complex applications, you should move HANG cells to a separate page. This allows you to easily route cells from different pages to the same HANG cell. By moving the HANG cells to a separate page, you improve the readability of the layout, since off-page connectors replace the connecting lines without cluttering the design.

Example of an application

The call flow diagram in Figure 4-2 represents a very simple application that answers a telephone call, plays a prompt to the caller, and then hangs up the call.

Figure 4-2: An example of a simple application



For purposes of visual clarity, some cell branches are left unconnected in the examples that follow. These unconnected branches typically deal with error handling. Branches are explained in depth in the “Branches” section, below.

Unconnected cells

If the application encounters any unconnected branches between cells, it will halt. Therefore, since it will not access further cells, the application will not play any prompts to explain the call flow to the caller. The application also will not perform cleanup functionality for the call.

Branches

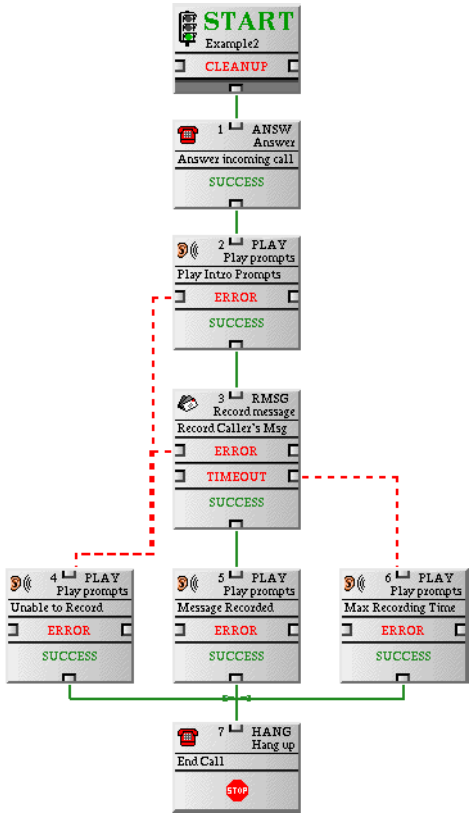
An application must be able to respond to abnormal results as well as normal results. For example, suppose an application asks a caller to leave a message. A cell named RMSG (an abbreviation for Record Message) can do this. However, you need to anticipate all the possible outcomes. The RMSG cell type has three branches to cover all the possibilities:

- The caller might successfully leave a message.

- There might be some sort of problem that prevents the system from recording the message.
- The caller might take too long to finish recording the message.

As shown in the following call flow in Figure 4-3, each of the three branches leads to a PLAY cell that plays a prompt appropriate to the situation.

Figure 4-3: An example of a simple application with branching



Buffers overview

AP software uses buffers to store and pass information during a call from one cell to another. You can store information in an OPEN IVR buffer as a string of any number up to 1920 ASCII characters (letters, numbers, spaces, or any other keyboard characters except commas). However, only 31 characters (plus one null termination byte) are entered into a buffer directly from an application. Refer to the *Symposium OPEN IVR Cell Catalog* Release 4.0, for additional information about the maximum buffer sizes that are supported for each cell.

For example, one cell in an application may ask a caller to enter a password by pressing the keys on the telephone. If the caller presses the 3, 7, 2, and 9 keys, the cell accepts the number 3729 and stores it in a buffer. Once the password is in a buffer, any other cell can use it. You might have a later cell check the password to see if it is valid.

The content of a buffer is always terminated by a null character. This is important to keep in mind when you write a user function.

You can use two different kinds of buffers when you are writing applications:

- system buffers that OPEN IVR software creates for you
- user-defined buffers that you create yourself

  Note: OPEN IVR is not case-sensitive to the names of user-defined buffers. If you name a user-defined buffer “digits” (all lowercase), OPEN IVR will use it as the “DIGITS” buffer.

As numbers, buffers support only integers (OPEN IVR supports signed integers from -2147483646 to 2147483646). STOR supports floating point numbers as a convenience, although the use is limited. For example, if you specify the number of leading and trailing zeros and blanks, you can use the `STRING` setting of the COMP cell to test a floating point number set in a buffer for equality.

You can store buffer information in the STOR cell. Buffers can contain information from another buffer, text string, or number.

- To enter data from another buffer into the source buffer, enter the name of the buffer without double quotes.

- To enter a text string into a buffer, enclose the ASCII characters in double quotes; for example, “text string”.
- To enter numbers into a buffer, enter the digit value without double quotes; for example, 35.

System buffers

A set of predefined system buffers is available to each application during each phone call. Many system buffers are automatically updated by cells, so you need to be careful that information you have placed in a system buffer is not accidentally replaced (for example, DIGITS and NUMBER OF DIGITS buffers are automatically updated by the MENU cell).

Furthermore, the buffer MESSAGE LENGTH is used by the RMSG cell and should not be used for other purposes. The CELL STATUS buffer is used by various cells to store status information and should not be used in application development.

A list of system buffers and an explanation of their typical uses follows in Table 4-1.

Table 4-1: System buffers

System buffer name	Usual contents
ANI DIGITS	Usually the phone number of the calling party, if available.
CELL STATUS	Used by various cells to store status information. Do not use this buffer in application development.
CURRENT MESSAGE	Contains the voice storage number where a message is read on the MRS.
DATA EXCHANGE #1-#5	Used to exchange information between both scheduled and called applications.
DATA EXCHANGE #6-#10	Used to exchange information between called applications.
DATE	Current date in the format MMDDYYYY.

System buffer name	Usual contents
DAY	Numeric form of the current day of the week, where 1 is Monday and 7 is Sunday.
DELIVERY ATTEMPTS	Number of times a delivery application attempts to make the delivery.
DELIVERY INTERVAL	Length of time between delivery attempts.
DELIVERY TIME	Scheduled date and time for delivery.
DELIVERY EVENT ID	Identification number for a specific delivery event.
DIGITS	Digits collected from callers. Also can contain the phone number of the called party, if available, and the category of telephone network data for retrieval.
FAX DATA	Name of the fax-data file.
ISWEB	If a new call is received from the World Wide Web, the buffer ISWEB is set to 1 to indicate that the application is executing through a request from the Web. The default setting for the buffer ISWEB is 0, which indicates that AP software receives a call from an MRS. ***Note:*** Do not write to this buffer.
LENGTH	Number of pages that are received for a fax.
MAILBOX ID	Identification number of a mailbox, which is a storage area for messages. The mailbox ID is often a phone number or extension.
MESSAGE LENGTH	Length of a message recorded in the RMSG cell. Do not use this buffer for other purposes.

System buffer name	Usual contents
NUMBER OF ANI DIGITS	Number of digits in the ANI DIGITS buffer.
NUMBER OF DIGITS	Number of digits in the DIGITS buffer.
NUMBER OF MESSAGES	Number of messages currently stored in a particular mailbox.
PASSWORD	Password associated with a mailbox.
PATH	Audio path ID.
RESULT	Result of a math operation.
SENDER ID	Identification of the message sender.
SQL ERROR CODE	Code returned by an access to a SQL DBMS.
SQL ROWS	Rows in an SQL table that match certain criteria.
TERMINATION DIGIT	Digit used to terminate user input.
TIME	Current time in the format of HHMM.

Accessing saved information from calls

The telephone network can pass certain telephone network information to an OPEN IVR application during either the initiation of an incoming call or the execution of an outgoing call by means of the outdialing cells (DOUT, BOUT, or COUT). (For more information about the DOUT, BOUT, and COUT cells and their parameters, see the *Symposium OPEN IVR Cell Catalog* Release 4.0.

> ***Note:*** Only certain types of telephone networks can pass information at call initiation. Contact your network provider for details on the service to which you subscribe.

Depending on the telephone network configuration, an incoming call can provide the following:

- the phone number of the calling party

- Called Line ID information
- call diversion codes
- the number from which the call is diverted

AP software stores this information until either the end of the call or the completion of an outdial, at which time this new information from the outdial overwrites the data stored during the initiation of the incoming call.

In addition, AP software will write the called number (DID digits) and calling number (ANI) to the DIGITS and ANI DIGITS system buffers, respectively.

Note: A single call can provide both DID and ANI information.

The exact number of digits received is defined by the trunk-type file and the number of digits the central office will pass across to the MRS. Refer to *Symposium OPEN IVR System Administration Guide Release 4.0.* The total number of characters that can be passed from the telephone network to AP software is 1920, but AP software can receive only 31 characters (plus one null termination byte) directly from an application.

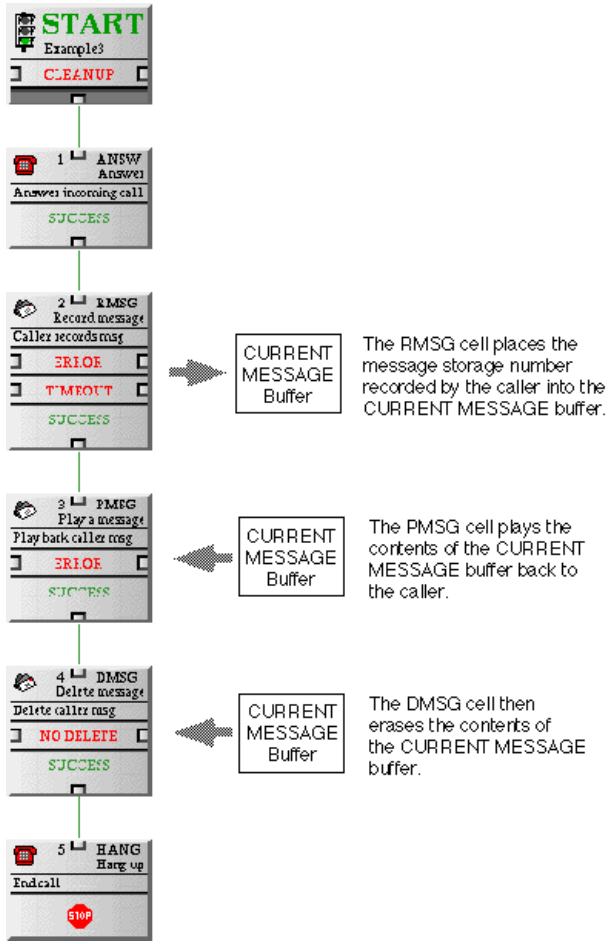
Depending on the telephone network configuration, an outgoing call may receive the following information:

- diversion reasons
- calling line category
- number from which the call was diverted

The CINFO cell can retrieve any of the supported data types from the digital telephone network for either incoming or outgoing calls by indicating the type of data for retrieval and the buffer for storage of the retrieved information. For more information about the CINFO cell, see the *Symposium OPEN IVR Cell Catalog Release 4.0.*

The call flow in Figure 4-4 shows an application that uses system buffers. This application answers a telephone call, records a message from the caller, plays the message back to the caller, erases the message from the MRS, and hangs up the call. In this call flow, each square represents a buffer. An arrow leading from a cell to a buffer indicates that the cell is storing information in the buffer. An arrow leading from a buffer to a cell indicates that the cell is using the information in the buffer.

Figure 4-4: Example of an application using buffers



User-defined buffers

OPEN IVR allows you to create as many as 200 buffers of your own in any one application at any point in the application. Once created, a user-defined buffer is available to all the cells in the application.

> ***Note:*** You can use a user-defined buffer only within the application in which you created it. You cannot pass information stored in a user-defined buffer to another application.

There are three reasons why you might want to create user-defined buffers:

- You find that your application needs many buffers, and there are not enough system buffers for your purposes.
- You want to have a buffer whose name describes its specific function in the application, and none of the system buffers seems appropriate.
- You want to have a buffer that is entirely under your control. Some cell types use buffers automatically. Check the cell type descriptions in the *Symposium OPEN IVR Cell Catalog Release 4.0* to see when and how this occurs. You may prefer to create a buffer that you know will not be used unless you specifically want it to.

A user-defined buffer can store up to 1920 characters, but can accept only 31 characters (plus one null termination byte) directly entered from an application. These characters can be letters, numbers, spaces, or any other keyboard characters. Refer to the *Symposium OPEN IVR Cell Catalog* for additional information about the maximum supported buffer size for each cell.

Databases

When you create an application, you may need to manipulate two kinds of information: messages and information. OPEN IVR has databases to help you manage and store this information.

Message databases

Message databases store messages that are left by callers and later retrieved for playback. Different applications have different requirements for storing and playing back messages. The messages may be either voice or fax messages.

For example, an IVR application might store recorded names and addresses for later transcription. A voice mail application will store messages so that they can be delivered to voice mail subscribers. The message database provides a generalized facility that is useful for these and other applications requiring message storage.

A message database consists of a set of mailboxes whose primary function is to associate a set of messages with a mailbox ID. The mailbox ID number is usually a phone number that will be dialed later by another caller, usually the owner of that mailbox, to retrieve the stored messages.

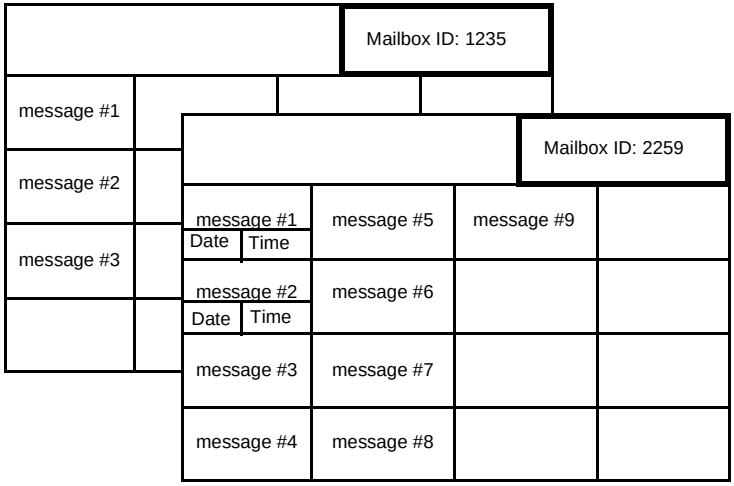
The number of mailboxes that you can use is determined by the number of significant digits associated with the mailbox ID (when you create a message database, you are prompted for the number of significant digits). For example, if the mailbox ID contains four significant digits, you can use up to 9999 mailboxes. Each mailbox can store up to 75 messages.

You can create many message databases, but each application can access only one. When you create an application, you specify the name of the message database in the START cell. A given message database can be used by more than one application.

> ***Note:*** Each message that is added to the database is date- and time-stamped.

The Application Editor provides a set of cells for accessing and updating messages in the database. Figure 4-5 illustrates the storing of messages in mailboxes. Mailboxes are explained in more detail later.

Figure 4-5: Mailboxes in a message database



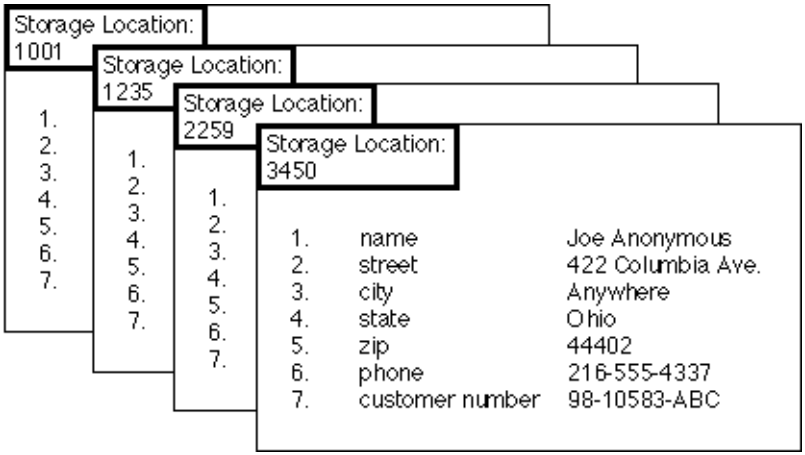
Information databases

An information database consists of a set of records, each of which stores strings of characters. You can create more than one information database, but only 20 information databases may be in use by all of the applications that are running at any one time.

To access the information, an application may have a CHEK (Check Database) cell to copy data from one record to a set of buffers you have chosen. Once the data has been placed in buffers, the application can use the data just as it would any other buffer data. For example, the application could look up a given record in a database, copy a phone number from the record to a buffer in the application, and then transfer the caller to the telephone number.

You can use the DINS, UPDT, or DDEL cells to insert, update, or delete records in your information database. Figure 4-6 illustrates records in an information database.

Figure 4-6: Records in an information database



The CHEK cell retrieves all defined fields in a record. Therefore, the appropriate number of buffers must be specified whether or not they are used.

The UPDT cell updates all defined fields in a record with the value placed in corresponding buffers (if there are not enough buffers of information to correspond to the number of fields updated, the contents of each of the extra fields defaults to the empty string). To avoid having incorrect values overwrite accurate information, ensure that the cell designates a buffer for every field updated.

Note: The maximum number of template buffers is 10.

Identification numbers for prompts and messages

All voice applications must manage two kinds of voice recordings: prompts and messages. Prompts are played to callers to guide them through applications and are generally used repeatedly. The application developer assigns an ID for each prompt.

Messages are created by callers and typically have a short life span. When a message is recorded, OPEN IVR automatically assigns a VSN number (voice storage number) to the message.

When you want to use prompts, you must know what prompt numbers to use. The system administrator, who is responsible for allocating ranges of prompt numbers for each application, can help you choose the correct prompt numbers for your application.

> ***Note:*** The MRS, operated by MRS software, has its own system for numbering voice recordings. The label assigned by the MRS is called a voice storage number (VSN), and is explained in the *Symposium OPEN IVR System Administration Guide Release 4.0.*

The AP software numbering system is different from the VSN system. AP software uses “handles.” Situations that require an understanding of the differences between these numbering systems are explained in this manual as they arise.

The utility `cnvrt` is used to calculate the handle number from a VSN, or the VSN from a handle number.

Outdialing

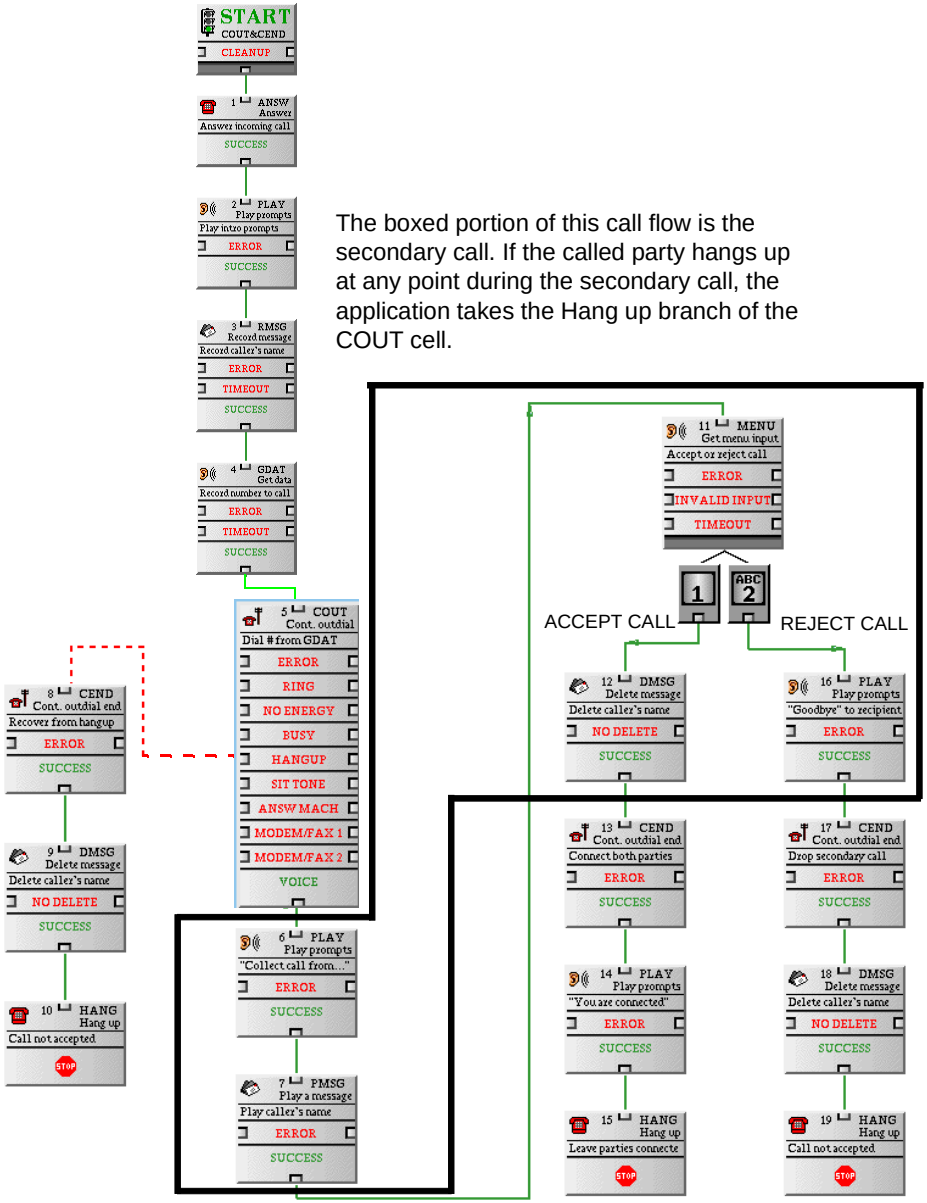


These cells provide outdialing, bridged outdialing (that is, two trunks: one in and one out), continuation outdial starting with a hook-flash, directed outdial for initiating calls, or dialing digits to a service or pager.

Example of an outdialing application

The application shown in Figure 4-7 demonstrates the use of the COUT and CEND cells. This application is intended to process a collect call. When the call is answered, the calling party hears introductory prompts and is asked to record his or her name. The caller uses DTMF keys to enter the number to be called. The application then makes a continuation outdial to that number, tells the called party that the calling party is making a collect call, and asks the called party if he or she wants to accept the collect call. If the called party accepts the collect call, the two parties are connected to each other. If the called party refuses the collect call or hangs up, the calling party is told that the call was not accepted and the call ends.

Figure 4-7: Application using the COUT and CEND cells



The boxed portion of this call flow is the secondary call. If the called party hangs up at any point during the secondary call, the application takes the Hang up branch of the COUT cell.

The call flow shown in Figure 4-7 is a simplified one; no error branches are shown. The call flow for the Collect Call application is explained in Table 4-2.

Table 4-2: Collect call application call flow

Cell	Explanation
Cell 1	An ANSW (Answer) cell answers the call.
Cell 2	A PLAY (Play Prompts) cell plays introductory prompts to the caller.
Cell 3	An RMSG (Record Message) cell records the caller's name.
Cell 4	A GDAT (Play Prompts and Collect Data) cell asks the caller to enter the telephone number that is to receive the collect call.
Cell 5	A COUT (Continuation Outdial) cell puts the calling party on hold and makes a secondary call to the number that the caller entered. If the secondary call is answered, Cell 6 is next. Error branches from this COUT cell are not shown. If the secondary call cannot be made or is not answered, the application goes to the appropriate error branch, takes the primary call off hold, and continues normally. The rest of the application could play prompts to tell the caller that the collect call could not be made, and then hang up. Note that Cell 5 has a Hang up branch. If the secondary call is made successfully but the called party hangs up during the course of the secondary call, the application must recover from the unexpected end of the secondary call. To do so, it stops what it is doing, no matter which cell is being executed at the time, and goes directly to the Hang up cell.
Cell 6	A PLAY (Play Prompts) cell plays the called party a prompt that says "You have a collect call from..." Cell 7 finishes the sentence.
Cell 7	A PMSG (Play a Message) cell plays the caller's name.

Cell	Explanation
Cell 8	This CEND (Continuation Outdial End) cell is the Hang up cell for the COUT (Continuation Outdial) cell. If the called party hangs up while the secondary call is in progress, the application immediately leaves the current cell and follows this branch. The Hook-Flash Parameter of this CEND cell is RECOVER. This allows the application to recover from the termination of the secondary call; it takes the calling party off hold.
Cell 9	DMSG (Delete Message) deletes the recording of the caller's name.
Cell 10	HANG (Hang Up) cell tells the first party that the collect call was not accepted, says good-bye, and hangs up.
Cell 11	A MENU (Play Menu and Get Data) cell asks the called party to make a choice to accept or reject the collect call. The caller indicates the choice by pressing a telephone key. If the call is accepted, the application branches to Cell 12. If the call is rejected, the application branches to Cell 16.
Cell 12	The collect call has been accepted; now a DMSG (Delete Message) cell deletes the recording of the caller's name. If the message is not deleted, it remains stored on the MRS even though it is unavailable to OPEN IVR after the application ends.
Cell 13	A CEND (Continuation Outdial End) cell connects the two callers and ends the secondary call.
Cell 14	A PLAY cell then tells the two callers that they are connected.
Cell 15	A HANG cell ends the call.
Cell 16	The collect call has been rejected; now a PLAY cell plays a prompt to the second party to say good-bye.
Cell 17	CEND (Continuation Outdial End) ends the secondary call. This CEND cell has been created with the Hook-Flash Type parameter set to DROP 2ND PARTY, which causes the application to end the secondary call and take the calling party off hold.

Cell	Explanation
Cell 18	DMSG (Delete Message) deletes the recording of the caller's name.
Cell 19	HANG (Hang Up) tells the first party that the collect call was not accepted, says good-bye, and hangs up the call.

Analog Display Services Interface (ADSI)

The Analog Display Services Interface (ADSI -- pronounced "add-zee") provides a better human interface than existing voice processing applications, since it eliminates the need to memorize menu choices. The caller can absorb the screen information while listening to voice overviews and instructions.

Your OPEN IVR application can use ADSI cells to interact with an ADSI display telephone as part of a call. By means of the ADSI cells, you can write an application that presents callers with a visual interface instead of or in addition to voice prompts. With these cells, you can also provide callers with the ability to navigate quickly through applications and to input information.

In a typical ADSI application, the MRS transmits information for display on a special, screen-based telephone. This special phone connects to the existing analog network service. The screen text can supplement or replace traditional voice prompt menus in the typical IVR application. The caller can respond to screen information by means of the telephone keypad or the softkey buttons (which are like application-programmable function keys).

In addition to DTMF keys and keys dedicated to a specific function (such as transfer to an operator), the ADSI phone has buttons for scrolling the screen and displaying information on the screen beyond the top or bottom of a current view. The caller can designate information on the screen for action by highlighting it with the active line indicator and pressing the <SELECT> key.

Applications that can be improved through the integration of ADSI include

- Host Interactive Voice Response features: Bank-by-Phone (where the display phone emulates an automated teller), catalog sales, order entry, and pay-per-view TV. As the display phone becomes ubiquitous in the residence, other lower-value services can be offered, such as a pizza-delivery order entry.
- Audiotex: stock market quotes, OAG information, sports, weather, and other services similar to those offered through Compuserve and Prodigy.
- Voice Mail advanced features: displayed message status and user directories.
- E-mail access.
- Automated Attendant features: menus of departments or people, visual options.
- Operator Services: electronic yellow and white pages.
- Telemarketing: “screen popups” of customer and product information provided to calling agents.
- FAX retrieval: accessed by a totally nonverbal application.
- Telecommunications for the deaf.

ADSI telephone

An ADSI telephone is a standard phone set with three additional components:

- A virtual display, some or all of which may be shown at any time on the simple character screen. This display consists of two pages:
  - — an information page, used to store data for the caller to scan
  - — a communications page, used for transient messages or data input

- A virtual softkey table providing a pool of 32 definable softkeys. The applications programmer can assign softkeys to each display line independently in sets of six keys, thereby allowing the flexibility for context-sensitive keys. An entry in the table contains
  - — a softkey definer to provide an index into the table
  - — a softkey label to provide a visual description of the functionality
  - — a return string to indicate the response to a key press by a caller
- Terminal control functionality composed of
  - — physical control functions to enable or disable the handset, keypad, and softkeys
  - — display control functions allowing the applications programmer to designate what information is presented, how it is displayed, the cursor control, and the method of scrolling through the virtual display

Designing an ADSI application

A combined IVR/ADSI application must intersperse alternate voice messages and data messages. The application must construct ADSI messages, format the screens, and define labels for both the softkeys and the tonal responses in terms of the DTMF strings.

ADSI cells accept character strings, buffers, and other parameters from their Parameters page. They format the information and transfer it as a VSN to the MRS, then transmit the ADSI databurst to the screen phone.

ADSI permits the transmission of either voice or data over the same line, although not simultaneously. Data transmitted to the screen uses existing network technology and touch-tone dialing. Every ADSI databurst is preceded by a CAS tone (the signal, or *chirp* tone, sent before an ADSI transmission to prepare the ADSI telephone to accept the data that is about to be downloaded from the MRS). At this point, voice transmission is momentarily muted and the keypad is temporarily disabled while the ADSI phone acknowledges the CAS tone with an ACK tone (a designated sequence of DTMF tones). This signals the MRS to commence data transmission, called a databurst.

When the databurst concludes, the phone transmits DTMF tones to acknowledge its receipt, and voice transmission resumes. The ADSI phone may need to do some processing after data transmission, but the application is unaffected.

The first databurst sent to an ADSI phone must be a Connect Session (dsCON cell). In order to complete the transmission properly, the final databurst must be a Disconnect Session (dsDIS cell).

If the MRS is connected to a telephone without display-based ADSI capabilities, the ADSI request will timeout waiting for an acknowledgment to the initial CAS tone. The MRS will then return a message to the ADSI cell indicating that the telephone is NOT ADSI. When writing an application, you should interpret the NOT ADSI branch as ADSI unavailable and adjust the call flow to proceed with a voice-only application.

> ***Note:*** If the dsCON cell returns NOT ADSI, all subsequent ADSI cells will automatically take their NOT ADSI branch as well. This allows you to place ADSI cells in your application without them interfering with voice-only calls.

Transmit ADSI data first, followed by any voice prompts, in order to start voice transmission in parallel with the display update. As the application is executed, the menu is automatically cleared before new text is written onto it. The dsINFO cell enables you to selectively either clear or overwrite the existing lines.

You can set every channel in the MRS as an ADSI-enabled channel. To do this, edit the trunk type file and enter the following line:

```
set hasadsi = 1;
```

Note: Parameter changes will go into effect after the system is reset.

You can run the ADSI process interactively (that is, `adsi -i`) to create permanent VSNs containing ADSI databursts. You can then download these databursts to the MRS using the PLF. Once installed, you can use these VSNs in any cell that allows a voice prompt (for example, the PLAY cell). This technique can improve ADSI transmission performance, since it allows you to create, store, and display a screen that is constant from call to call. Design your application to act on the outcome of the dsCON cell to determine whether or not to play these VSNs.

ADSI (ds) cells

The OPEN IVR ADSI cells include the following:

- dsCON—Connect Session
- dsDATA—Data Input Forms
- dsDIS—Disconnect Session
- dsINFO—Display Information
- dsINPUT—Get Data
- dsKEYS—Define Soft Keys
- dsMENU—Display Menu
- dsMODE—Select Voice or Data Mode
- dsUSER—Connect ADSI User Function

You interweave voice prompts and ADSI transmissions in your OPEN IVR application in order to implement a well-defined user interface to the ADSI phone.

OPEN IVR's ADSI cells provide the capability to

- Establish a connection to an ADSI display telephone.
- Define sets of softkeys common to menu or information displays.
- Format capability for display on the ADSI screen.
- Present visual menus on the display screen where
  - — The caller can use scroll buttons in order to move through the list of item choices, highlight a choice, and press the SELECT button

to choose it.

— The caller can press a DTMF key corresponding to one of the numbered selections in order to choose that selection.

— The caller can press one of the labeled softkeys in order to perform other application functionality, such as go back one menu level, go to the top of the application, transfer to a service representative, get help with the current menu, get general information about the application. These buttons are application-specific.

- Obtain numeric input from the caller by presenting a data format within which the caller can supply information. The display can also contain softkeys to allow the caller to edit the data before it is sent to the application.
- Store gathered—information in a buffer for later use by the application.
- Display information on a screen. The caller can use the built-in scrolling buttons to peruse the data and, when done, press a labeled softkey (provided by the application) to perform more functions. These buttons are application-specific.
- Choose between Voice and Data mode.
- Update specific lines on an information display.
- Connect to a User Defined function created to allow access to specific low-level commands supported by an ADSI phone.
- Terminate the connection to the ADSI display telephone.

ADSI cells are detailed in the *Symposium OPEN IVR Cell Catalog, Release 4.0.* A brief description of each of these cells follows.

dsCON—Connect Session

dsCON establishes a communications session with an ADSI telephone. Execute this cell before other OPEN IVR ADSI cells in the call flow so the other cells can send ADSI parameters to the display telephone.

The dsCON cell allows the user to connect to the phone for feature download. After connecting for feature download, the user can use the dsUSER cell to download scripts to the ADSI phone.

Enabling Feature Download

To build an ADSI application that enables Feature Download, follow these steps.

1. In the dsCON cell, set the value of the parameter `Connect For Feature Download` to `Yes`.
2. Execute dsUSER cells that create and send databursts containing ADSI feature download messages to the ADSI phone.
3. Execute dsDIS cells where the parameter `Connect For Feature Download` is set to `Yes`.

dsDATAdData input forms

dsDATA downloads a form containing textual prompts and one or more input format patterns to an ADSI telephone. It also defines the keys "Clear," "Backspace," and "Accept."

dsDIS—disconnect session

dsDIS terminates a communications session with an ADSI telephone.

dsINPUT—get data

dsINPUT decodes an ADSI-encoded string and places the alphanumeric data in a buffer.

dsKEYS—define soft keys

dsKEYS downloads a list of softkey definitions to an ADSI telephone. Subsequent dsINFO and dsMENU cells in the application can then use these definitions.

dsMENU—display menu

dsMENU downloads a list of menu options to an ADSI telephone. The application can define up to nine selection items and up to six softkeys.

dsMODE—choose voice or data mode

dsMODE enables you to choose between Voice and Data mode. When you specify Data mode, the handset and the keypad of the ADSI phone are disabled.

dsINFO—display information

dsINFO downloads a list of ASCII text lines to an ADSI telephone. This cell directs an application to output information to the caller.

dsUSER—connect ADSI user function

dsUSER provides the ADSI application with the capability to access specific low-level commands supported by an ADSI phone.

Application Programming Interface (API) for issuing level 3 ADSI commands

Since AP software does not provide cells to accommodate accessing specific low-level commands supported by the ADSI phone, the application developer can create customized user functions using the ADSI "C" Application Programming Interface (API) to accomplish this.

These user functions are accessed from your application with a dsUSER cell. This way, ADSI level 3 messages (calls from the API) can be created using the User Function, which can wrap them up into ADSI level 2 messages and send them to the ADSI phone.

> ***Note:*** The dsUSER cell is designed for use by experienced programmers. Before attempting to write a user function using the API, refer to Chapter 8, "Creating User Functions," of this guide.

Constructing an application

When you are building your application, you should consider constructing your application as a set of interconnected pieces or "modules." Each module performs a specific task that can be easily identified and changed if necessary without inadvertently affecting other aspects of the application. The Application Editor contains cells that allow you to subdivide your application into task-specific modules.

Modules can be reused by more than one application. For example, consider a common function that prompts for an account number and Personal Identification Number (PIN) and then verifies them. One approach is to let every application implement this functionality. But for consistency and reuse, it would be more efficient for the application to share a common module to accomplish this task.

Modules may be internal to an application file and accessible only to that application. Modules may also be external and accessible by other applications. It is typical to have minor (internal) functions reside within the same application file as the application that calls it, and to have major functions and shared modules reside in a separate application file, thereby facilitating access by many applications.

System buffers are passed between external modules, while user-defined buffers are reinitialized every time an external module is entered. This means, for example, that the data in the DIGITS buffers contains whatever information in it when a module was called. A buffer called “account number” on the other hand will not contain anything until explicitly set in the called module.

Building applications

To create any application follow these steps. The steps are broken into a series of procedures.

1 Design an application.

Before you begin using the ApplicationEditor, plan your application. Think about how you want to modularize your application.

- What are the major functions of the application?
- What are the minor functions of the application?
- What are the common functions of the application?
- What are the database requirements of the application?
- What is the best tool to satisfy this need— message, information, or SQL?
- Are any external interfaces required (e.g., 3270, VT-100, custom)? Plan how you will get and transfer the data from this external interface.

2 Open the Application Editor

Open the Application Editor to use it for creating the application.

3 Create an Application File

You must create a new file to store your application. Each application is stored on the application server by naming and saving the file.

4 Edit the Start cell

Once a file is created, begin work on the application by editing the values of the default cell and setting global parameters for your application.

5 Select cells

Select the cell types that make up the application. Choose cells that accomplish the action you want carried out at each step.

6 Customize cells

Once you have selected a cell, name and customize each one by setting the necessary parameters. Many parameters initially contain default values.

7 Select prompts

Check to see if the prompts you need have already been recorded and stored. If necessary, you can compose and record new prompts.

8 Install and run an application

Once you have saved the application, install and run it to see how it performs.

9 Test and debug

After you create your application, thoroughly test it to make sure it performs as desired in all situations.

10 Edit an application

If you need to correct or update the application, you can reopen and edit it.

Creating a development directory

Before using the Application Editor, create a directory in which to develop your applications. To do this, follow these steps.

1 Change directory (cd) to the home directory where AP software is installed.

2 At the shell prompt, type the following command:

`mkdir directory_name` <Enter>

Where `directory_name` can be any directory name you choose.

You will probably want to create your development directory subordinate to the base directory of AP software.

For example, to create a development directory named `devel_app`, change the directory to the home directory of AP software, and then type at the shell prompt:

`mkdir devel_app` <Enter>

Note: Although not required, Nortel recommends that you create a separate development directory for your applications. In order to run your developed application, copy your application from the development directory to the applications directory.

Automatic Speech Recognition using the LISTEN cell

The LISTEN cell performs Automatic Speech Recognition (ASR) for large vocabularies. It can initiate the recognition process, retrieve information spoken by a caller, evaluate and assign a confidence level to it, and store the information in a buffer, making the spoken information available to the application.

The LISTEN cell facilitates applications that require large vocabularies, rely on vendor-specific attributes from different packages at different points in the application, or depend on vendor-specific functionality.

This cell can direct an ASR application to perform any of the following ASR functions:

- Evaluate multiple results with associated confidence levels.
- Activate a sub-vocabulary.

- Support DTMF functionality.
- Format returned results.
- Direct rejected responses.
- Report exceptions if the response is too quick, too short, too loud or too soft.

Note: For more information on ASR, refer to the *Symposium OPEN IVR Speech Recognition Option Guide*.

Evaluating multiple results with associated confidence levels

The LISTEN cell provides the capability to associate a score (as a percentage) with an utterance to determine the probability that the utterance matched a word or phrase in an active sub-vocabulary. This score is called the Confidence Level.

By coding an ASR application to access the information placed in the `Table of Results` and `Table of Confidences` on the Parameter's page of the LISTEN cell and comparing confidence levels of the results (or subset of results), the application can evaluate the utterance. The application can then direct the call flow of the application according to the outcome of the comparison.

Supporting DTMF functionality

There are two modes of DTMF handling: `Data` and `Control`. The LISTEN cell contains a parameter called `DTMF Used For`, which sets the mode of DTMF functionality to either `Data` or `Control`, as appropriate.

`Data` mode (which can be either speech or DTMF) directs the application to store a response in a buffer that the application can access when needed. For speech, this response is stored in the buffer specified in the `Table of Results`; for DTMF, the digit collected is stored in the DIGITS buffer. The LISTEN cell also indicates in the INPUT METHOD buffer (from the `Method of Input Detected` parameter) whether the response is DTMF (a value of 1) or speech (a value of 0). The application can access this buffer to determine which branch the call will take out of the cell.

`Control` mode (which is always speech) allows the caller to control the playing of a prompt. By pressing the DTMF keys specified in the Default cell, the caller can perform functions such as PLAY speed-up and PLAY slow-down, or abort the action of a cell. The `Method of Input Detected` parameter will always be set to speech (0) when the call flow exits the LISTEN cell from the Control mode.

Note: All DTMF keys are defined in the Default cell. These values are passed to the LISTEN cell.

Formatting returned results

An application can return ASR-specific information formatted by one of two methods. An ASR vendor-dependent method can automatically format the information, or OPEN IVR—specific features can be employed to provide customized formatting.

The following options are available:

- FULL—returns all of the information reported.
- STRING—returns the text of recognized information.
- TAGS— returns a list of specific key words or phrases from the recognized utterance.
- INDEX—returns a table offset that indicates a unique ID directing the application to the recognized utterance.
- TIME/DATE—returns a standard structure containing the date and/or time information formatted properly, regardless of the manner in which it was spoken.

Note: Refer to your vendor's documentation for a list of supported formats. If a format is specified that is not supported by the vendor, AP software sets the EXCEPTION STATUS buffer to 1000.

Directing rejected responses

A rejected response is not necessarily an error, and therefore requires a unique response from the ASR application. A rejection is a recognition result created when an application analyzes an utterance and does not find a match in the associated vocabulary. To accommodate this condition, the LISTEN cell has a REJECTION branch to direct the call flow appropriately.

Reporting exceptions

The LISTEN cell includes a status buffer on the Parameter's page to report the following error conditions that can occur during the recognition attempt:

- response too quick
- response too short
- response too loud
- response too soft
- format not supported

OPEN IVR stores this information in a global buffer, EXCEPTION STATUS. The application can use this information to direct the call flow if it is affected by the response to the various error conditions.

> ***Note:*** If the application does not detect speech, or if the caller speaks more than the time allotted, a timeout condition results.

Logging audited information

ASR-specific information can be gathered and saved with either an ASR vendor-dependent method or IVR Developer—specific cell auditing techniques. *Symposium OPEN IVR System Administration Guide, Release 4.0,* "Creating Reports" chapter, provides more information on generating reports.

Chapter 5: Building a sample application

This section includes information about building a sample application and using the Graphical Debugger to debug an application.

Developing a sample application

In this chapter, step-by-step instructions lead you through creating a sample application that you can load and run. By following the lesson presented here, you will gain experience that will help you design and build applications of your own.

These are the steps you will follow:

- designing the application
- selecting prompts
- creating the application file
- editing the START Default cell
- placing and naming the cells
- customizing the cells
- saving and naming the application
- reviewing the application
- loading and running the application
- editing the application

This chapter explains only the rules needed for constructing this particular sample application.

Designing the application

Designing the sample application is easy—it has already been done for you. This application, called LEAVEMSG (see Figures 5.1 through 5.4,) allows a caller to leave a message that will be saved in the Message database. This application will not actually put the recorded message into a mailbox unless you choose a mailbox that has been created previously by some other application. However, the LEAVEMSG application will still run and perform all of its other intended activities. This application is designed to run on DID trunks.

Figure 5-1: Page 1 of the LEAVEMSG application

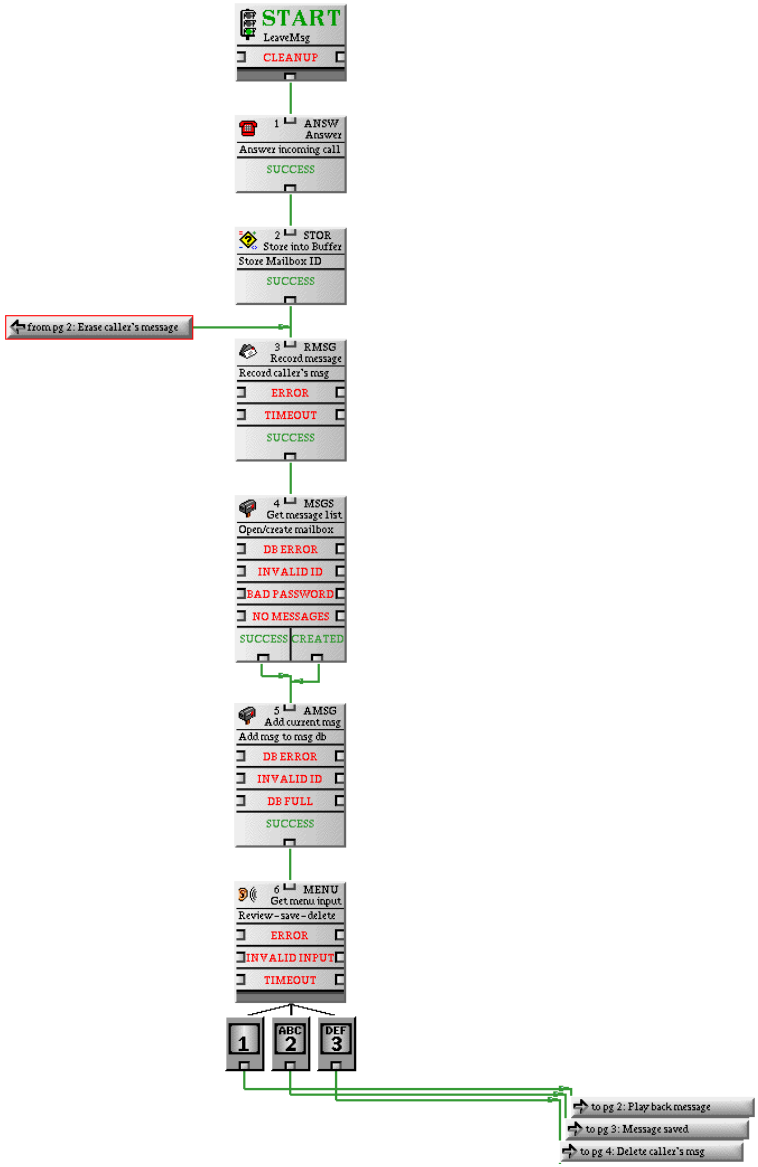


Figure 5-2: Page 2 of the LEAVEMSG application

LeaveMsg
Page 2

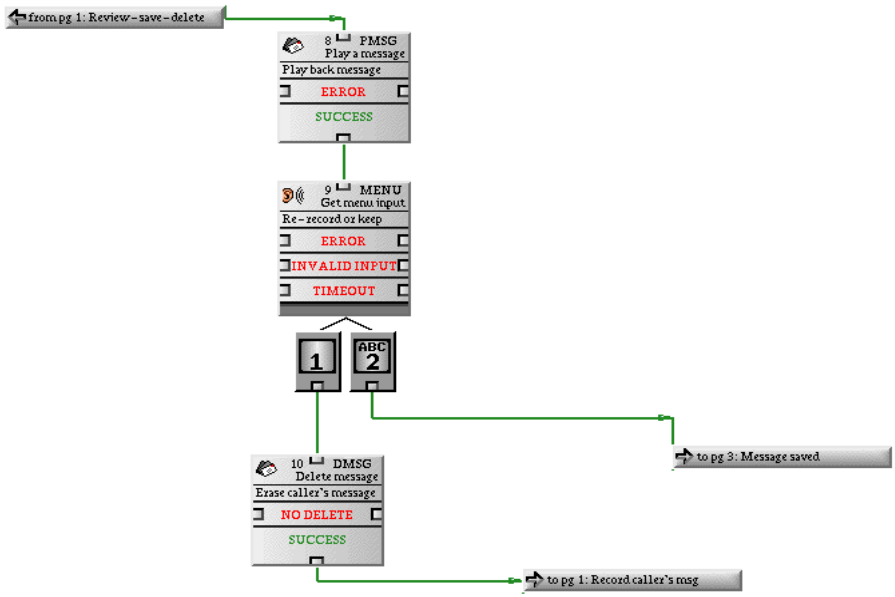


Figure 5-3: Page 3 of the LEAVEMSG application

LeaveMsg
Page 3

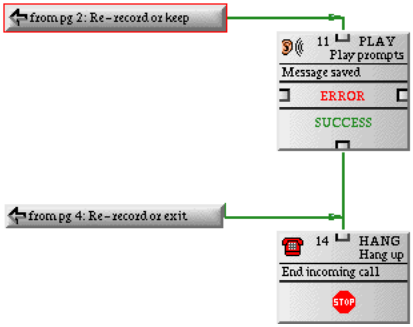
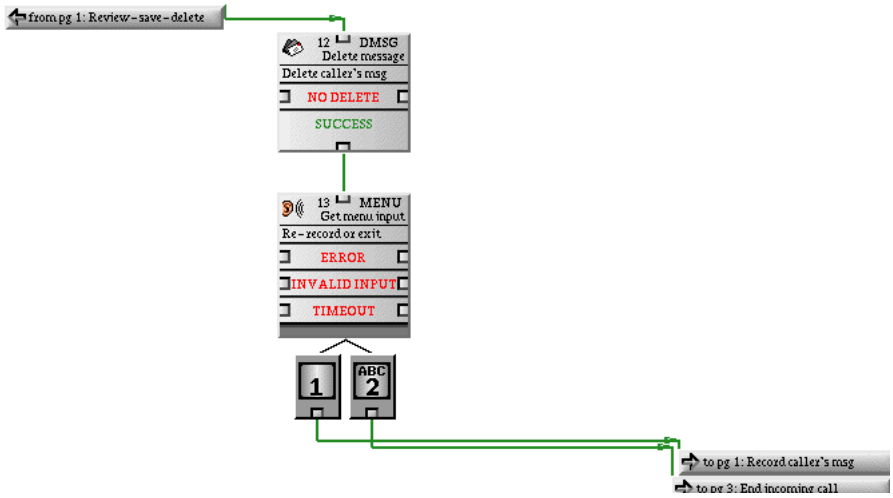


Figure 5-4: Page 4 of the LEAVEMSG applications



This simple example covers just the basic functions needed by a message-gathering application. It has no provision for responding to unexpected results. Once you have followed the tutorial to gain some experience with creating applications, you can use this sample application and the applications in Chapter 5 as starting points to create other, more involved applications. The following information describes a sample call flow.

Call flow for LEAVEMSG

Figures 5-1, 5-2, 5-3, and 5-4 show the call flow for the LEAVEMSG application. Here is a cell-by-cell explanation.

Before the first cell in the application is executed, DID digits are received and stored in the DIGITS buffer.

Table 5-1: LEAVEMSG call flow

Cell number	Cell type	Description
Cell 1	ANSW	The application begins by answering the call.
Cell 2	STOR	The application takes the digits that are in the DIGITS buffer (that is, the number that was called) and copies the digits to the MAILBOX ID buffer. These digits will be used to identify the mailbox that will store the message.
Cell 3	RMSG	The caller hears a prompt that says, “At the tone, leave your message. When finished, press pound.” The caller speaks the message and then ends the recording process by pressing the # (pound) key on the telephone. This cell records the message.
Cell 4	MSGS	Opens the mailbox identified by the DID digits collected by the STOR cell (cell 2). If the mailbox exists, then an exception branch to “No Messages” is taken. If a mailbox with this identification number does not exist, this cell creates one.
Cell 5	AMSG	Adds the caller’s message to the message database in the mailbox.
Cell 6	MENU	The caller is presented with a menu: “To review, press 1. To save, press 2. To delete, press 3.” The caller presses a digit to make a menu selection. The application branches to cell 8, 11, or 12, depending on the menu selection.

If you select 1 to review the message, the following call flow is executed:

Cell number	Cell type	Description
Cell 8	PMSG	The application plays the message back to the caller.

Cell 9	MENU	The application plays another menu. This time, the choices are to rerecord the message or continue. If the caller selects rerecord, the application branches to the next cell. If the caller selects continue, the application branches to cell 11.
Cell 10	DMSG	If the caller selects rerecord from in the previous MENU cell, this cell erases the message, then sends the call back to the RMSG cell (cell 3) so the caller can rerecord the message.

If you select 2 to save the message, the following call flow is executed:

Cell number	Cell type	Description
Cell 11	PLAY	Since the caller's message is already in the appropriate mailbox, no further action is necessary. This cell tells the caller that the message has been saved. Cell 14 is the next cell.

If you select 3 to re-record the message, the following call flow is executed:

Cell number	Cell type	Description
Cell 12	DMSG	This cell deletes the original message.
Cell 13	MENU	The caller hears another menu. The choices are rerecord or continue. If the caller chooses rerecord, the application branches back to cell 3. If the caller chooses continue, the application branches to the next cell.
Cell 14	HANG	The application plays a prompt that says, "Thank you. Good-bye," and the application hangs up.

Selecting prompts

Prompts are vitally important to your applications because they are the means by which you give information to your callers and guide them through the steps they should perform, such as leaving a message in a mailbox or making a selection from a menu. AP software has a collection of standard prompts that are commonly used in applications; they are listed in Appendix A. You will not have to record any prompts because LEAVEMSG uses only the standard prompts that follow

Table 5-2: LEAVEMSG prompt number descriptions.

Prompt number	Description
32	“pound”
55	“Thank you. Good-bye.”
59	“At the tone, leave your message. When finished, press...”
60	“Your message has been delivered.”
66	“To save, press 1. To review, press 2. To delete, press 3.”
67	“To rerecord, press 1. To continue, press 2.”

Create the application with the Application Editor

AP software’s default application directory is the `apps` directory. Since the OPEN IVR auto-save feature frequently saves your application, Nortel recommends that if the directory in which you want to store your application is not the directory `apps`, then first navigate to the directory in which you want to keep your application.

Changing directories

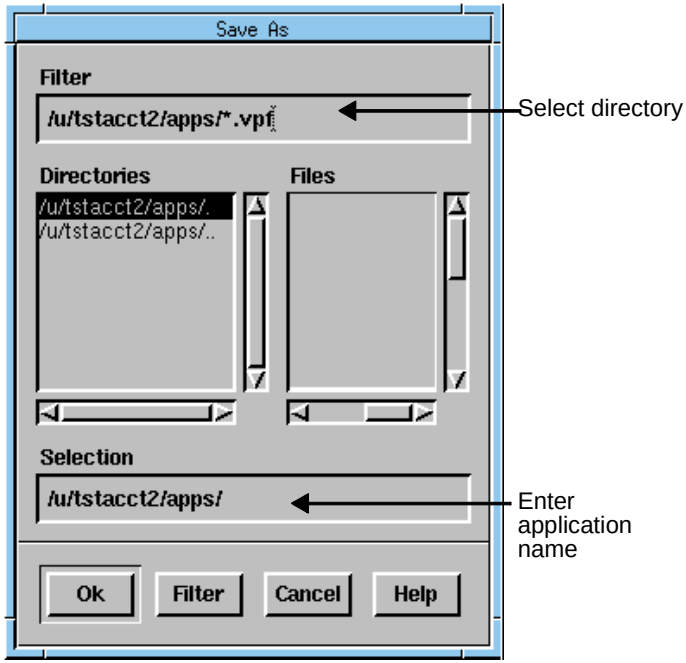
To change the directory, follow these steps.

1 Open the Application Editor. From the File menu, select “Save as...” The Application Browser appears as shown in step 4.

2 Navigate to your development directory. Use the Filter function or enter the path name directly.

3 Name the application. Enter a name for your application in the Selection text field.
The name will appear on the START cell.

4 Once you have entered your application name, click the Ok button on the Application Browser.



Editing the START (Default) cell

Once you have created your application file, you can edit the values in the START cell screen. In the START cell, you can

- Select a message database for the application.
- Establish default values for some of the parameters in cells throughout the application.
- Establish the Minimum Voice Duration.
- Determine which cell in the application will be executed first.
- Establish the Cleanup Handler.
- Establish the type and name of the SQL database.

  Note: The START cell is not actually a cell (it has no effect on the actual call flow). Rather, it performs functions that affect the cells of the application.

For our sample application, only the Message Database Name parameter needs to be changed. There is no need for you to change the default values for parameters in this application. In addition, you will not be using the Voice Confirmation feature of the MENU cell, so there is no need to select a value for the Minimum Voice Duration. Cell 1 will be executed first in the LEAVEMSG application, and cell 1 already appears in the Next Cell field, so there is no need for you to make a change. Finally, the application will not have a cleanup handler, so the value of the Cleanup Handler Cell parameter can be left as it is.

Changing the START cell Message Database Name parameter

The only required change to the Sample Application START cell is to change the Message Database Name parameter.

To make this change, follow these steps.

1 Open the Start Cell by double clicking with MB1 on the START cell in the Application Editor window.

The Application Default cell appears.

2 In the Message Database field, type the new name, **`practice`**.

3 On the START cell, click Apply.

The message database name will become **practice**.

Note: To use a message database, you must first create it. Refer to Chapter 6, “Using message databases,” and Chapter 7, “Using information databases,” for directions on creating a database.

Placing and naming the cells

The next step is to place all of the cells in the application and assign a name to each one. The cells used in the LEAVEMSG application are listed in Table 5-3. The tutorial takes you step-by-step through the process of creating the first two cells, after which you can easily create and name the remaining cells.

Table 5-3: Cells in the LEAVEMSG application

Cell number	Cell type	Cell description	Name
Cell 1	ANSW	Answer call	answer call
Cell 2	STOR	Store	save mailbox ID
Cell 3	RMSG	Record message	record message
Cell 4	MSGS	Get Message List	open/create mailbox
Cell 5	AMSG	Add current message	save message
Cell 6	MENU	Play Menu and Get Data	save-review-delete?
Cell 8	PMSG	Play message	play message back
Cell 9	MENU	Play Menu and Get Data	rerecord or continue?
Cell 10	DMSG	Delete message	erase message
Cell 11	PLAY	Play prompts	say message sent
Cell 12	DMSG	Delete Message	erase message
Cell 13	MENU	Get menu input	rerecord or exit
Cell 14	HANG	Hang up	end call

Placing and naming the ANSW cell (cell 1)

To create the ANSW cell, follow these steps.

1. Click the ANSW cell in the cell palette with MB1.
2. Click beneath the START cell in the drawing board with MB2. The ANSW cell (untitled) appears on the Application Editor Drawing Board.
3. Double-click on the ANSW cell.
4. In the first field of the ANSW cell parameter window, enter "`answer call`".
5. Click Apply to confirm your entry.

   The ANSW cell ("`answer call`") appears on the drawing area with the new name.

Creating and naming the STOR cell (cell 2)

To create the STOR cell, follow these steps.

1. In the cell palette, click MB2 on the STOR cell.
2. In the drawing board, click MB2 beneath the ANSW cell.

   The STOR cell (untitled) appears in the drawing area of the Graphical Application Editor window.
3. Double-click the STOR cell to open and then name the cell.
4. To name the cell, click in the first field in the STOR cell parameter window and type "`save mailbox ID`".
5. Click Apply to confirm your entry.

   The STOR cell ("save mailbox ID") appears on the drawing area with the new name.

Placing and naming the remaining cells

Repeat the process of choosing cell types and names until you have created all of the cells in this application. See Table 5-4.

Table 5-4: Remaining cells for the LEAVEMSG application

Cell number	Cell type	Name
Cell 3	RMSG	record message
Cell 4	MSGS	access mailbox
Cell 5	AMSG	save message
Cell 6	MENU	save-review-delete?
Cell 8	PMSG	play message back
Cell 9	MENU	rerecord or continue?
Cell 10	DMSG	erase message
Cell 11	PLAY	say message sent
Cell 12	DMSG	delete caller's message
Cell 13	MENU	rerecord or exit
Cell 14	HANG	end call

Customizing the cells

You're now ready to customize all the cells you have just created. You will be changing the values of the parameters and connecting to the next cells and tables to meet the specific needs of this application. We will go through each cell individually.

Customizing the ANSW cell (cell 1)

The ANSW cell type has no parameters and no tables. If you refer back to the call flow diagram in Figure 5-1, you see that the next cell for answer call should be STOR. You need to connect the ANSW cell to the STOR cell.

1 Click MB1 on the connector at the bottom of the ANSW cell.

The pointer changes to a pencil cursor.

2 Point the pencil cursor on the connector at the top of the STORE cell, and click MB1 to form a source—destination pair.

The Application Editor creates the connection. The next cell for the ANSW cell is now the STOR cell.

Customizing the STOR cell (cell 2)

There are no tables for this cell, but the parameters can be modified if necessary. The STOR has two parameters:

- Source (Buffer or Value)
- Destination Buffer

The purpose of this STOR cell is to store digits from the DIGITS buffer in the MAILBOX ID buffer. Therefore, the Source Buffer will be the DIGITS buffer and the Destination Buffer will be the MAILBOX ID buffer.

Connecting the next cell

As you can see from the call flow diagram (Figure 5-1), the next cell is RMSG. In the next several steps, you will change the parameters and connect to the next cell.

1 Double-click the STOR cell to open its Parameters window.

2 Select the field that says “Source (Buffer or Value).”

3 Select a name for this field by clicking the push button to the right of the field.

a Click this button to display the “Select A Buffer Name” browser, where a list of valid values appears.

b From this list, select “DIGITS.”

c Click Apply.

The browser disappears and “DIGITS” replaces the initial value.

Note: When you click the browser and select a buffer, type the name only once. Thereafter, access the selection in the windows environment to avoid inadvertently typing an incorrect name and creating new buffers in the system.

4 Select the parameter that says “Destination Buffer.”

5 Change the setting from the initial value, “DIGITS,” to “MAILBOX ID.”

a Select the name of the buffer by clicking the push button to the right of this field.

b Click this button to display the “Select A Buffer Name” browser.

c From the list, select “MAILBOX ID.”

d Click Apply.

The browser closes and “MAILBOX ID” replaces the initial value.

6 At the bottom of the STOR cell parameter window, click Apply.

Your changes are saved and this window closes.

7 The next cell after the STOR cell is the RMSG cell. Connect these two cells in the same way as for the ANSW cell.

Customizing the RMSG Cell (Cell 3)

There are a few more features of the editor that you should try. To learn about these additional features, start modifying the RMSG cell.

For this application, the only parameter you need to change is the “# of Introduction Prompts.” This cell will play two introduction prompts.

1 Double-click the RMSG cell.

The record message cell pop-up appears.

As you see on the screen, this cell has many parameters. If you want, you can turn to Chapter 12 and read the descriptions of the parameters, next cells, and tables.

The RMSG cell type has three branches to next cells (Error, Timeout, and Success). In this application, all of the branches lead to AMSG cell.

The RMSG cell type has a “Number of Prompts” field where you enter how many introduction prompts the record message cell plays. Following this field are additional fields where you specify which prompt will play in the order shown.

Note: The “Number of Prompts” field is set automatically when you select prompts to play.

2 Move the cursor to field 1.

3 Enter the prompt number 59, “At the tone, leave your message. When finished, press...”
Prompt 59 starts a sentence that tells the caller how to leave a message.

4 Move the cursor to field 2.

5 Enter the prompt number 32, “pound.”
This next prompt will tell the caller to press the termination digit. In this cell, the # key is the termination digit.

6 Press Apply at the bottom of the RMSG cell parameter window.

Your changes are saved and this window disappears.

7 Connect all branches from the RMSG cell to the AMSG cell.
Connect these branches in the same way you did above.

Customizing the remaining cells

Now that you have customized the first three cells, you know how to do the same for the remaining cells. You don’t have to work on them in any particular order—you can go back to any cell whenever you choose. Changes for the remaining cell follows.

Note: You can accept the default parameter values in the MSGS cell.

Customizing the AMSG cell (cell 5)

Parameters	There is only one parameter. Don't change the initial value.
Next cells	Connect this cell to the first MENU cell "save-review-delete?" The next cell for every branch is the first MENU cell.

Customizing the first MENU cell (cell 6)

Parameters	The Number of Valid Inputs should be changed to 3.
Tables	There are two tables: the Play Prompt Table and the Valid Input Table.

The Play Prompt Table has one entry: 66 (this prompt says, "To save, press 1. To review, press 2. To delete, press 3.")

The Valid Input Table has three entries:

- Select digit 1 for Valid Input: 1 Next Cell: 9 (say message sent).
- Select digit 2 for Valid Input: 2 Next Cell: 6 (play message back).
- Select digit 3 for Valid Input: 3 Next Cell: 11 (delete message).

Next cells	The next cell for every error branch will be the HANG cell. The next cell for menu choice 1 is the PLAY cell. The next cell for menu choice 2 is the PMSG cell. The next cell for menu choice 3 is the DMSG cell.

Customizing the PMSG cell (cell 8)

Parameters	There is no need to change any of the initial values.
Next cells	The next cell for both branches is the second MENU cell, “rerecord or continue?”

Customizing the second MENU cell (cell 9)

Parameters	The # of Valid Inputs should be changed to 2. There are two valid inputs (menu choices) for this cell and they have to be listed in the Valid Input Table. There is no need to change any of the other parameter values.
Tables	There are two tables: the Play Prompt Table and the Valid Input Table.

The Play Prompt Table has one entry: 67. (This prompt says, “To rerecord, press 1. To continue, press 2.”)

The Valid Input Table has two entries:

- Select digit 1 for Valid Input: 1 Next Cell: 8 (erase message).
- Select digit 2 for Valid Input: 2 Next Cell: 9 (say message sent).

Next cells	The next cell for the Error, Invalid Input, and timeout branches is the HANG cell. The next cell for menu choice 1 is the DMSG cell. The next cell for menu choice 2 is the PLAY cell.

Customizing the DMSG cell (cell 10)

Parameters	There are no parameters.
Next cells	The next cell for both branches is Cell 3, RMSG.

Customizing the PLAY cell (cell 11)

Parameters	There is no need to change any of the initial values.
Next cells	The next cell for both branches is the HANG cell.
Tables	There is one table: the Play Prompt Table. It has one entry: 60. (This prompt says, “Your message has been delivered.”)

Customizing the DMSG cell (cell 12)

Parameters	There are no parameters.
Next cells	The next cell for both branches is the HANG cell.
Tables	There are no tables.

Customizing the HANG cell (cell 14)

Parameters	The Number of Hang-up Prompts should be changed to 1. This cell plays one hang-up prompt, which will have to be listed in the Hang-up Prompt Table.
Next cells	There is no next cell because this is the last cell in the application.
Tables	There is one table: the Hang-up Prompt Table. It has one entry: 55. (This prompt says, “Thank you. Good-bye.”)

Saving the application

Once all the cells in the application have been customized, the application needs to be saved.

To save your application, follow these steps.

1 If you have already named your application, select “Save” from the File menu.

Your application is saved.

2 If you are naming your application, select “Save as” from the File menu.

a Type the name that you are assigning to your application at the end of the path in the selection field. Do not type the extension .upf when you enter the name; the extension is added by AP software software when you click OK.

b Once you have entered the name for your application, click OK.

Your application is saved under the assigned new name. Then the graphic-based editor’s main window is displayed.

Reviewing the application

To review the application, follow these steps.

1 Select “Open” from the File menu.
The application directory appears.

2 Select “LEAVEMSG” from the list of files in the scroll area.

3 Click Apply.

The application appears on the drawing board of the graphical application editor’s window.

This enables you to easily step from cell to cell to review your work. As you look through the cells, you can use the techniques you’ve already learned to correct any errors you see.

Loading and running the application

Once you have reviewed the application, you are ready to load and run it for testing. However, before you load the application, you must copy it to the directory called `apps`. This is the only directory from which AP software can load applications. You should leave a copy of the application in your development directory so that you can use it as the basis for building the next version of the application.

> ***Note:*** The LEAVEMSG application is designed to be run on channels that are configured as DID trunks.

For instructions on copying, loading, and running the application, see Release 4.0 of the *Symposium OPEN IVR System Administration Guide* or consult your system administrator.

Editing the application

After you have tested an application, you may want to make changes to it. When you want to modify an application that has already been installed, remember to edit the copy that is in your development directory, not the running copy in the `apps` directory.

You can edit an application in much the same way that you create a new application. When you are editing, you can create new cells. You can also modify any existing cell by changing its name, type, parameters, tables, and next cells.

Debugging an application

IVR Developer's Application Editor includes the Graphical Debugger. Application developers can use this graphical tool to identify and modify problems within a selected application, channel, or combination of an application with a particular channel. You can use the debugger to interrupt an active application and to step through the cells within an application. As you navigate through the execution of the application, you can stop at specified cells to view the contents of the buffers. The Graphical Debugger allows you to control specific channels of an application and leaves other channels assigned to an application available.

ATTENTION

The Graphical debugger is designed to debug an application on one channel or debug a specific channel on a *development* system. It was not intended to be used on a *production* system.

Note: If you invoke an application in Debug mode, you can only modify buffer contents; you cannot modify cell parameter pages.

Overview

When you invoke an application in Debug mode, the menu option Debug is added to the Application Editor menu bar. Figure 5-5 displays the Application Editor menu bar before you invoke an application in Debug mode.

Figure 5-5: Application Editor menu bar



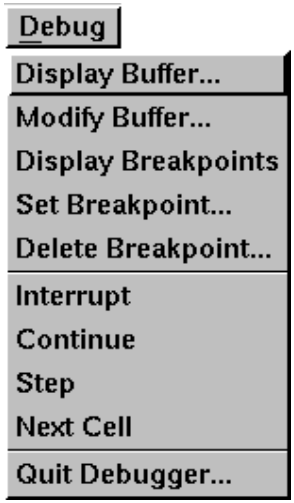
Figure 5-6 displays the Application Editor menu bar in Debug mode.

Figure 5-6: Application Editor menu bar in Debug mode



The Debug pull-down menu lists the options available to an application in Debug mode. You can use these options to initiate and control the execution of an application, as well as to debug an application. Figure 5-7 displays the Debug pull-down menu.

Figure 5-7: Graphical debugger menu options



You can control the execution of the application by using breakpoints or by interrupting the execution of the application at any cell. When the application is interrupted or reaches a breakpoint, the Application Editor scrolls to and highlights the selected cell.

Existing Application Editor status messages are displayed in the Status window, but after four seconds the Application Editor displays status messages in the Status window for the debugger functions Interrupt, Continue, Step, Next Cell, and Quit. These messages are useful when you want to go to another page to view a cell, because the debugger message remains in the Status window so that you know the location of the cell where the execution of the application was either interrupted or reached a breakpoint.

Once the execution of the application is stopped at a cell, you can view and modify the contents of buffers in order to verify and change the information collected and passed by other cells. If you identify problems within an application that are related to the call flow or the parameter settings of specific cells, then you can quit the debugger and edit the call flow and cell parameter pages in the Application Editor.



CAUTION

Risk of service interruption

The Graphical Debugger is designed to debug an application on one channel or debug a specific channel on a *development* system. Do not use the graphical debugger on a *live* (production) system.

Invoking the Graphical Debugger

Depending on the command line arguments that you provide, you can Debug a particular application, a specific channel, or a combination of an application with a specific channel.

Note: You need to load an application before you can invoke it in debug mode. Use the Symposium OPEN IVR Control Panel's Application Management tools to load applications. Refer to Release 4.0 of the *Symposium OPEN IVR System Administration Guide* for more information about loading applications.

Debugging an application

To debug an application, type the following command line at the prompt in the system shell UNIX window:

```
xae -debug app_name
```

Where ***app_name*** is the name of the application that you intend to debug.

Note: When you invoke the debugger in conjunction with an application, enter only the name of the application itself. Do not enter the path name for the application.

When you invoke an application in Debug mode, the first call into any channel assigned to that application activates the debugger. Use this invocation if you are interested in debugging an application, regardless of which channel handles the incoming call.

Debugging a channel

To debug a channel, regardless of which application is assigned to that channel, type the following command:

```
xae -channel channel_number
```

Where ***channel_number*** is the number of the channel that you intend to debug.

Invoking the debugger is useful when you suspect that a particular channel is malfunctioning.

As soon as the first call comes into the specified channel, the application that is assigned to that channel becomes active and the debugger controls the channel. Application Editor highlights the START cell and stops the execution of the application there because you may not know what application is assigned to that particular channel when it first becomes active. This behavior is true for all three methods of invoking the debugger.

Debugging an application with a specific channel

To invoke an application in Debug mode in conjunction with a specific channel, type the following command:

```
xae -debug app_name -channel channel_number
```

Where ***app_name*** is the name of the application and ***channel_number*** is the number of the channel to invoke in debug mode.

This method of invoking the debugger is useful for debugging applications that are not working correctly only when they are assigned to certain channels. In this example, the specified application is invoked in Debug mode when a call comes into the specified channel.

If you assign 10 channels to an application and you invoke that application in Debug mode, the graphical debugger controls one of those 10 channels. The other nine channels are available to the application. In this case, only the channel that receives the call into the application is in Debug mode, and the graphical debugger allows the application to handle calls into any of the other nine channels.

Controlling the execution of an application

You can control the execution of an application and determine the way the debugger proceeds through an application, including the following:

- setting breakpoints
- displaying breakpoints
- deleting breakpoints
- using the options Interrupt and Continue to control the execution of an application
- using the options Step and Next Cell to control the execution of an application

  Note: You can open only one debug window at a time. For example, you need to close the Set Breakpoint window before you can open the Breakpoint Information window.

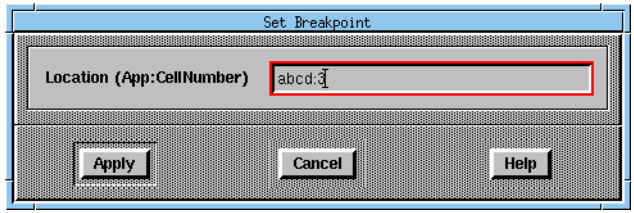
Setting breakpoints

You can use breakpoints to stop the call flow of an application at a specific cell. When the call flow of an application reaches the selected cell, the execution of the execution is stopped and the cell is highlighted. Once the selected cell is highlighted, you can display and modify the buffer contents. You can view the parameter settings of the cell by double-clicking with MB1 on the cell.

Refer to the section “Modifying buffers” on page 5-31 for information about viewing and modifying the contents of the buffers used by the cell.

1 To set a breakpoint, select the option Set Breakpoints... from the Debug menu.

The Set Breakpoint window is displayed.



In this example, the breakpoint is defined as `abcd:3`.

2 Use the following format to enter breakpoint information:

`App:CellNumber`

Where `App` is the name of the application and `CellNumber` is the number of the cell. In the example shown above, the name of the application is `abcd` and the number of the cell is `3`.

You can also enter only the cell number. If you enter only the cell number, 3 in this example, the debugger automatically fills in the name of the application that is currently displayed on the screen. It is helpful to specify the application name when there is more than one application involved with the execution of the application that is currently displayed on the screen; for example, when subroutines are invoked by GSUB cells.

You can perform the following functions in the Set Breakpoint window:

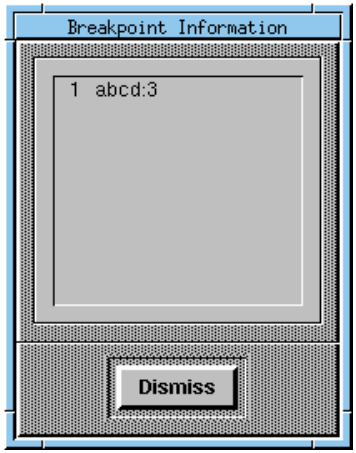
- Click the Apply button to use the specified application and cell number as a breakpoint.

- Click the Cancel button to close the window. Any values entered in the Location field are not set as breakpoints.
- Click the Help button to display the Set Breakpoint online Help window.

Displaying breakpoints

You can display the current breakpoint settings by selecting the option Display Breakpoints from the Debug menu. Figure 5-8 shows the Breakpoint Information window.

Figure 5-8: Breakpoint Information window



The Breakpoint Information window displays currently set breakpoints. The format of the information listed in the window is as follows:

```
Breakpoint_Number App:CellNumber
```

Where `Breakpoint_Number` is the number of the breakpoint, `App` is the name of the application, and `CellNumber` is the number of the cell that is set as a breakpoint.

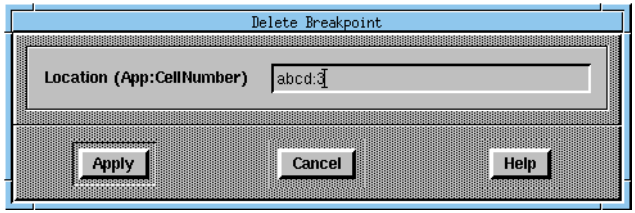
In Figure 5-8, the breakpoint number is `1`, the application name is `abcd`, and the cell number is `3`. If you click the Dismiss button, the Breakpoint Information window closes.

Note: You cannot modify the information in the Breakpoint Information window.

Deleting breakpoints

To delete breakpoints, follow these steps.

1 To delete currently set breakpoints, select the Delete Breakpoint... from the Debug menu.
This displays the Delete Breakpoint window.



2 Type the name of the breakpoint in the correct format, and click the Apply button.

Use the Delete Breakpoint window to perform the following functions:

- Click the Apply button to delete the specified breakpoint.
- Click the Cancel button to cancel any modifications made during this session.
- Click the Help button to display the Delete Breakpoint Help window.

Using the options Interrupt and Continue to control the execution of an application

You can interrupt and continue the execution of an application at any cell within the application in order to view the cell parameters and to modify the buffer contents.

Use the Interrupt and Continue options to perform the following functions:

- Use the Interrupt option at any point when the application is running. For example, if you use the Interrupt option to stop at a cell, then the Application Editor stops the execution of the application.
- Use the Continue option to continue the execution of the application from the point at which it is either interrupted or reaches a breakpoint.

Using the menu options Step and Next Cell to control the execution of an application

You can use the options Step and Next Cell to control the execution of an application from the point at which the execution is either interrupted or reaches a breakpoint.

- If you click the Step option, then the Application Editor proceeds to and highlights the next cell, including GSUB cells that link subroutines to the application the that is currently displayed on the screen.
- If you click the Next Cell option, then the Application Editor proceeds to the next cell in the application. Note that if you use the Next Cell option the debugger does not stop at cells within a subroutine unless those cells are specified as breakpoints. A delay may occur while the application executes the subroutine until the next cell is reached in the application that is currently displayed.

  Note: If you use the Next Cell option, the application is interrupted at a cell in a subroutine only when a breakpoint is already set, or if you select the option Interrupt during the execution of the subroutine.

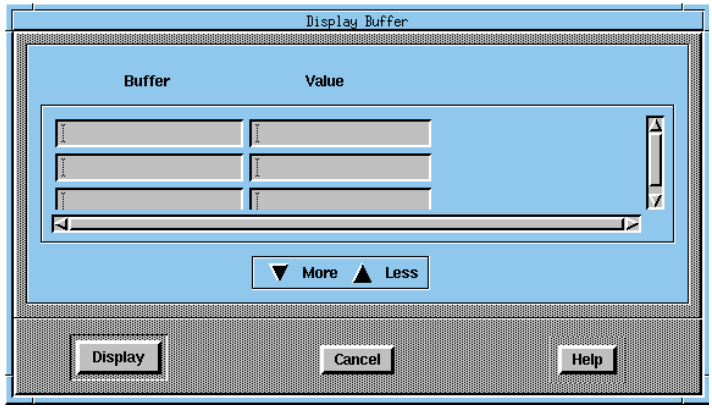
Modifying buffers

You can display and modify the contents of the buffers used by a cell when the execution of the application reaches a breakpoint or if it is interrupted.

Displaying buffers

You can display the contents of the buffers used by a particular cell in an application. To display buffers, follow this step.

1 Select the option Display Buffer... from the IVR Developer Debug menu.
This displays the Display Buffer window.



The Display Buffer window contains two columns, `Buffer` and `Value`. In the example shown above, the DIGITS buffer contains the digits 1, 2, and 3. The NUMBER OF DIGITS buffer contains the value 3.

You can perform the following functions with the Display Buffer window:

- The default setting for the Display Buffer window is three buffers. Click the “More” arrow to display one more row of information. Click the “Less” arrow to display one less row.
- If you enter the name of a buffer that you wish to display in the Buffer field and click on the Display button, the system displays the current value for that buffer.
- If you click the Cancel button, the system closes the Display Buffer window.
- If you click the Help button, the system displays the Display Buffer online Help window.

The Display Buffer window automatically corrects the spelling and capitalization of buffer names as you type them. For example, if you enter the system buffer name digits, the Display Buffer window corrects the capitalization to DIGITS.

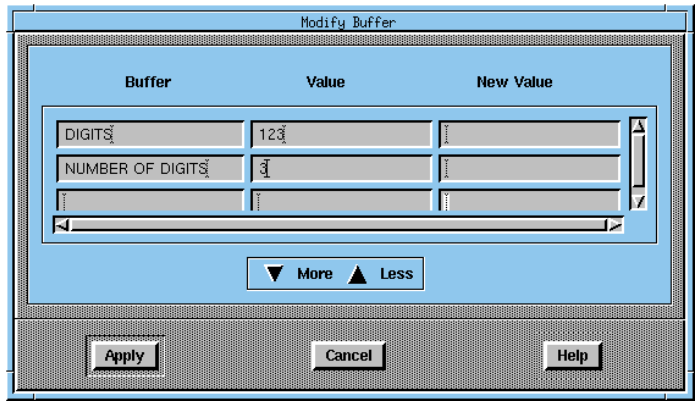
You can also leave rows empty and maintain the ability to display the contents of the remaining buffers. For example, if you select three buffers and then delete the middle entry, you do not need to adjust the order of the remaining buffer; the debugger skips over the blank space and displays the contents of the next buffer in the list.

> ***Note:*** You cannot modify the information in the Display Buffer window. To modify the contents of a particular buffer, select the option "Modify Buffer..." from the Debug menu.

Modifying buffers

You can modify the value of a buffer to determine whether the new value enables the application to execute successfully. To modify a buffer, follow these steps.

1 Select the option Modify Buffer... from the Application Editor Debug menu.
The Modify Buffer window is displayed.



The difference between the Modify Buffer window and the Display Buffer window is the additional column `New Value`. To modify the contents of a buffer, type the new value of the buffer in the column `New Value`. The system continues to display the original value in the column `Value`. Use the Modify Buffer window to perform the following tasks:

- The default setting for the Display Buffer window is three buffers. Click the “More” arrow to display one more row of information. Click the “Less” arrow to display one less row.
- Click on the Apply button to save any modifications to buffer contents. For example, if you modify the value of a buffer and then click the Apply button, these changes are applied to the application.
- Click the Cancel button to cancel any modifications made during this session.
- Click the Help button to display the Modify Buffer online Help window.

You can also leave rows empty and maintain the ability to display the contents of the remaining buffers. The debugger skips over the blank row and allows you to modify the contents of the next buffer in the list. The application developer can modify the contents of buffers for an application in debug mode using the option Modify Buffer. You can delete one row of information by clearing the buffer name entry in that row.

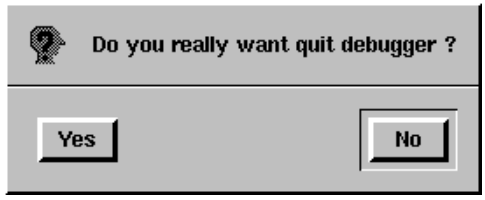
> ***Note:*** The buffers you enter in the Display Buffer window are also the buffers displayed in the Modify Buffer window.

Quitting the Graphical Debugger

To quit the debugger, follow this step.

1 Select the option “Quit Debugger...” from the Debug pull-down menu.

The quit dialog box is displayed.



The default setting is No. If you click the No button, the current application remains in Debug mode.

If you click Yes, the system executes the rest of the application and relinquishes the channel from the debugger. The channel is available to the application for call processing.

You need to properly quit the active debugger session before you quit the Application Editor. If you select Quit from the File menu, the same dialog box is displayed, but the channel used by the debugger for that particular application execution is still controlled by the debugger and the application is still running in the background.

If you select “Quit Debugger...” from the Debug menu, the current debugger session is closed, but the Application Editor does not exit and you can open and edit other applications.

Chapter 6: Using message databases

AP software has an extensive database facility that can be very useful when building applications. AP software databases can hold two basic information types:

- messages recorded by callers
- data, in the form of strings of characters, that can be used as part of the application

Although both types of information are maintained using the same database facilities, AP software makes a distinction between the databases that it can create. A message database is used to store messages, and an information database is used to store data.

This chapter explains how to create and use message databases.

Message databases store messages that are left by callers and later retrieved for playback. Different applications have different requirements for storing and playing back messages. For example, an interactive voice response application might store recorded names and addresses for later transcription. A voice mail application may store messages so that they can be delivered to another caller. The message database provides a generalized facility that is useful for these and other applications requiring message storage.

A message database consists of a set of mailboxes and a set of cell types for accessing and updating the messages in message databases. The primary function of the message database facility is to associate messages with a number, usually a phone number that will be dialed later by another caller to retrieve the stored messages. Each mailbox is labeled with its own mailbox ID number for easy retrieval. See Figure 6-1.

Figure 6-1: An example of a mailbox in a message database

			Mailbox ID: 2259
message #1			
message #2			
message #3			
message #4			

You can have more than one message database within OPEN IVR, but there may be only one message database per application. When you build an application, you put the name of its message database on the Default Cell screen. A message database may be used by more than one application.

Applications typically add messages to the message database during the course of a telephone call. These messages are collected in mailboxes. The mailbox can be identified by the DID number that was collected from the caller during the course of the call. The mailbox ID number can be four or five digits long.

The number of available mailboxes is determined by the number of significant digits associated with the message database. AP software prompts you for this information when you create the database through the Database Editor. For example, a message database that uses four significant digits to specify mailbox IDs can have up to 9999 mailboxes. Each mailbox can store up to 75 messages. When the messages within a mailbox are retrieved, they can be retrieved in the order of either newest message first or oldest message first. The retrieval order is determined by the GMSG (Get Message) cell.

Time stamp

When a message is added to a mailbox by an AMSG (Add Current Message) cell, it is stamped with the current date and time. When the message is later retrieved from the mailbox by a GMSG (Get Message) cell, the date and time are also retrieved and stored in buffers so that they are available to the application. The application could have PDAT (Play Prompts with Data) cells to read the date and time to the caller so the caller will know when the message was left.

Password

The application can limit access to the database by assigning a password to each mailbox. This ensures that confidential messages can be retrieved only by authorized persons. The password is assigned to the mailbox when a MSGS cell is used to create the mailbox. The password can be changed with the UPWD (Update Password) cell type. If passwords are not required, they can be omitted.

Message database cell types

You can use the following cell types to handle messages and the message database.

RMSG (Record Message)

Records a message onto the MRS, but does not store it in the message database.

AMSG (Add Current Message)

Adds a message previously recorded by RMSG to a mailbox in the application's message database.

MSGS (Get Message List)

Creates and/or accesses a mailbox, or is used to check the existence of a mailbox. When used to create a mailbox, the MSGS cell assigns a password to the contents of the PASSWORD buffer. A mailbox must be created before messages can be added to it. A mailbox must be accessed before messages can be retrieved from it.

GMSG (Get a Message)

Retrieves a message from a mailbox after the mailbox has been accessed with MSGS.

PMSG (Play a Message)

Plays a message to the caller.

CMSG (Copy Message)

This cell type copies a message from one MRS to another so that applications running on the other MRS can have access to it.

DMSG (Delete Message)

Removes a message from a mailbox, erases the message from the MRS where it has been stored, and updates the file `vsn`*x*`.data` (*x* is the MRS number) in AP software.

UPWD (Update Password)

Changes the password associated with a particular mailbox.

UMSG (Update Message List)

Updates a mailbox by confirming the changes made to it by DMSG and AMSG cells. UMSG is no longer needed because every application automatically updates the message database whenever AMSG or DMSG is executed.

Applications and the current message

The concept of the current message is the key to understanding how applications handle messages. Any application that processes messages works on one message, the current message, at a time for the caller.

When a new message is recorded with an RMSG (Record Message) cell, it becomes the current message. If a message was previously recorded and stored in a message database, it becomes the current message again when it is retrieved with a GMSG (Get a Message) cell.

The AMSG (Add Current Message), PMSG (Play a Message), CMSG (Copy Message), and DMSG (Delete Message) cell types all operate on the current message.

This scheme for handling messages simplifies the process of building applications. Because an AP software application works on only one message at a time, there is never any need for you to consider the actual message number. The message number is stored in the “CURRENT MESSAGE” buffer.

Opening the Information Database Editor

To open the Information Database Editor, follow these steps.

1 Click MB1 on the Database tool button, as shown below.



Alternative by start the Information Database Editor from the Control Panel to start it from the system shell. To do this,

a Change the current directory to `exe` in the base directory of AP software.

b If you are in an X environment, type at the shell prompt

```
sde_start <Enter>
```

c For non X environments, type

```
sde <Enter>
```

Whether you access the Information Database Editor from the Control Panel or the shell prompt, OPEN IVR displays the Information Database Editor main menu



2 Move the cursor to the item you wish to select, and press **<**Enter**>**. The keys work as follows:

<Up Arrow>	Moves the cursor up one line.
<Down Arrow>	Moves the cursor down one line.
<Left Arrow>	Moves the cursor to the top of the page.
<Right Arrow>	Moves the cursor to the bottom of the page.

Creating and naming the database

To create and name a new database, follow these steps

1 From the Information Database Editor main menu, press F3 for “New Database.”
The following prompt appears:

```
Enter the Name of the Database:
```

2 Type a name. A message database name can be up to eight characters long and can include any characters except spaces and commas.

3 When you have finished typing the name, press <Enter>.

Choosing the number of significant digits

This procedure continues from step 3 above.

1 The following prompt appears:

```
Enter The # Of Significant Digits (between 4 and 5,
inclusive):
```

2 Select the number of significant digits for this database. For a message database, the number of significant digits identifies the mailbox. You can type either 4 or 5 and press <Enter> to enter the new value, or press <Enter> to accept the default value of 4.

Note: Be careful when you choose the number of significant digits because, once the number has been chosen, it cannot be changed.

Choosing the database type

1 Enter an **M** for a message database when prompted, and press <Enter>. The following prompt appears:

```
Press any key to continue.
```

2 Press any key on the keyboard.
You are returned to the original screen.

Using message databases

To use message databases, follow these steps.

1 Create a message database using the Database Editor.

2 Create a mailbox using the MSGS cell by selecting the parameter for creating a mailbox.

3 Let callers add messages to the mailbox.

4 Let the owner of the mailbox retrieve the messages.

Following is a more detailed explanation of these steps.

Creating mailboxes

An application may have an MSGS (Get Message List) cell type to let the caller create a new mailbox and assign it a password; the mailbox is then available for message storage and retrieval. The message that is assigned should either be stored in the application or the application should use a GDAT cell to let you input the password.

Later, when a message-retrieval application uses a MSGS cell to access the mailbox, the password must be provided if a password is assigned when the application is created. If it is necessary to change the password, an application can use the UPWD (Update Password) cell type.

Note: You can create mailboxes without assigning passwords.

Adding messages

An application may have an RMSG (Record Message) cell to allow the caller to record a message and then an AMSG (Add Current Message) cell to add it to a mailbox.

Note that one cell records the message and another stores it in the message database. Often, a caller hangs up immediately after leaving a message and, normally, an application stops immediately if the caller hangs up before the application finishes. However, AP software has a special feature to ensure that a newly recorded message will be saved in the message database even if the caller ends the recording process by hanging up.

If the application has an AMSG cell as the "Success" Next cell for the RMSG cell, the application will not end until AMSG has saved the message. You can also place an AMSG cell in the cleanup handler that is executed at the end of every call.

Retrieving messages

Before an application can retrieve messages, it must use an MSGS (Get Message List) cell to access the mailbox. In addition to accessing the mailbox, MSGS counts the number of messages in the mailbox and stores that number in a buffer for later use by the application. Once the mailbox has been accessed, the application can use a GMSG (Get a Message) cell to retrieve one message; then PMSG (Play a Message) can play it back to the caller, CMSG (Copy Message) can copy it to another MRS, and DMSG (Delete Message) can delete it.

An application can use a single GMSG (Get a Message) cell in a loop to retrieve every message in a mailbox. This is useful when the owner of the mailbox wants to go through the messages. One of the parameters in GMSG determines the retrieval order, which may be oldest first or newest first. Along with each message, GMSG also retrieves the date and time that the message was added to the message database and then stores this information in buffers. The application can use PDAT (Play Prompts with Data) cells to read the date and time to the caller.

Examples of applications that use a message database

The call flow diagrams in this section show activity in a message database. The four applications together make a very simple voice mail system. Note that these examples are simplified—they have no branches for handling errors—but they do illustrate the basic concepts of how a message database is used. These four applications all use the same message database; its name must be entered on the Default Cell screen for each application.

Not all of the prompts used by these applications are standard prompts, so if you want to build these applications yourself, you will have to record some additional prompts. For recording procedures, see the chapter "Managing Prompts on the MRS" in the *System Administration Guide* or ask your system administrator. You would also have to create the databases for these applications before using them.

These four applications are summarized in the following sections.

An application to create mailboxes

Figure 6-2 shows an application called MakeMailbx that lets the caller create one or more mailboxes in the message database that the other three applications will use. It also lets the caller assign a password to each mailbox; however, it is not necessary to assign a password. This application is intended to be used only by the administrator of the voice mail system.

An application to present a menu

Figure 6-3 shows an application called MsgMenu. It asks the caller to make a choice either to leave a message or to retrieve messages, then turns the call over to another application, depending on the caller's choice.

Before this application begins, the MRS receives DID digits identifying the mailbox and stores them. They are used to identify the mailbox used by both LeaveMsg, which lets a caller record a message, and RetrievMsg, which lets the owner of the mailbox retrieve the stored messages. LeaveMsg is intended to be used by the general public, while RetrievMsg is intended to be used only by subscribers to the voice mail system.

An application for leaving a message

Figures 6-4 to 6-7 shows LeaveMsg, the application activated by MsgMenu if the caller wants to leave a message. LeaveMsg records the message and then stores it in the mailbox. Note that although the mailbox has a password, the caller does not need to use it when leaving a message.

An application for retrieving messages

Figure 6-8 shows RetrievMsg, the application activated by MsgMenu if the caller wants to retrieve messages. After entering the correct password, the caller can retrieve the messages, play them back, and then save or discard each one. Password protection ensures that only the owner of the mailbox can retrieve the messages.

Note: These applications could have been constructed as a single application instead of four. For the purpose of modular application design, it is easier to add features at a future time if the application is designed as separate modules.

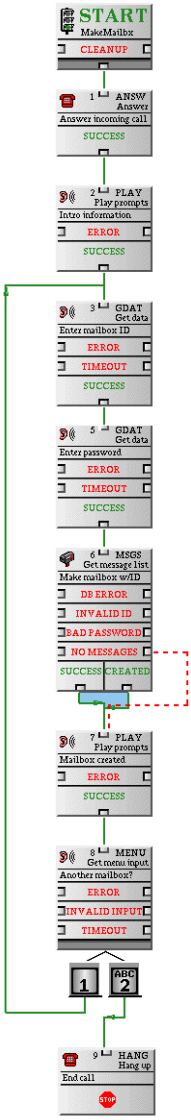
MakeMailbx application

The MakeMailbx application creates mailboxes. It asks the caller to enter a mailbox ID number and a password and then uses that information to create a mailbox. The caller has the option of creating more than one mailbox.

Table 6-1: Cell descriptions.

Cell number and type	Cell description
Cell 1 (ANSW)	Answers the incoming call.
Cell 2 (PLAY)	Plays the prompt that greets the caller.
Cell 3 (GDAT)	Asks the caller to enter a mailbox ID number. Accepts and stores the number in the MAILBOX ID buffer.
Cell 4 (GDAT)	Asks the caller to enter a password for the mailbox. Accepts and stores the password in the PASSWORD buffer.
Cell 5 (MSGS)	Creates a mailbox using the buffers MAILBOX ID and PASSWORD.
Cell 6 (PLAY)	Plays the prompt that tells the caller that the mailbox is created.
Cell 7 (MENU)	Plays a menu to the caller: “To create another mailbox, press 1. To quit, press 2.” Accepts the choice from the caller and then branches accordingly.
	Menu Selection 1: Create another mailbox. The application returns to Cell 3 (GDAT).
	Menu Selection 2: Quit. The application branches to Cell 8 (HANG), which says good-bye and hangs up.
Cell 8 (HANG)	Hangs up and ends the call.

Figure 6-2: Call flow of the MakeMailbx application



MsgMenu application

Figure 6-3 shows the MsgMenu application. The MsgMenu application simply finds out whether the caller wants to leave a message or retrieve messages, and then uses an EXEC cell to turn the call over either to the LeaveMsg application or to the RetrievMsg application. If the new application cannot be started for any reason, MsgMenu plays the caller a prompt to explain the problem and then hangs up.

Before MsgMenu begins, DID digits are received and stored in the DIGITS buffer.

Cell 1 (ANSW)	Answers the call.
Cell 2 (STOR)	Copies the digits from the DIGITS buffer to the MAILBOX ID buffer. The contents of this and all other system buffers will be passed to the next application when it is started by the EXEC cell.
Cell 3 (MENU)	Plays a menu to the caller: “To leave a message, press 1. To retrieve messages, press 2.” It then accepts the caller’s menu choice and branches accordingly.

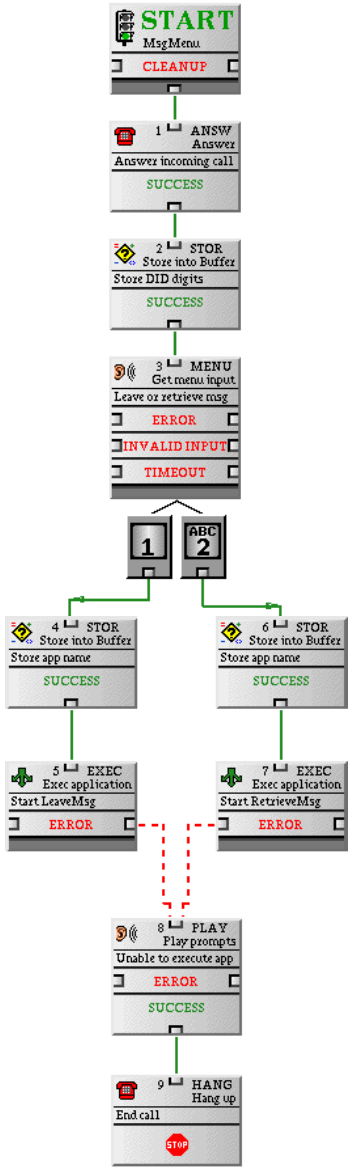
Menu selection 1: leave a message

Cell 4 (STOR)	Stores the name of the application in the user-defined buffer, app_name.
Cell 5 (EXEC)	Tries to start the LeaveMsg application. If it succeeds, control of the call is turned over to LeaveMsg and the MsgMenu application ends. Otherwise, go to Cell 8.

Menu selection 2: retrieve messages

Cell 6 (STOR)	Stores the name of the RetrievMsg application in the ANI DIGITS buffer.
Cell 7 (EXEC)	Tries to start the RetrievMsg application. If it succeeds, control of the call is turned over to RetrievMsg and the MsgMenu application ends.
Cell 8 (PLAY)	If LeaveMsg or RetrieveMsg cannot be started, Cell 8 tells the caller that there is a problem.
Cell 9 (HANG)	Says good-bye and hangs up.

Figure 6-3: Call flow of the MsgMenu application



LeaveMsg application

Figures 6-4, 6-5, 6-6, and 6-7 show the LeaveMsg application, which allows the caller to leave a message in a mailbox. LeaveMsg begins when the MsgMenu application turns over control of the call with an EXEC cell.

> ***Note:*** The mailbox ID is the DID number that was received by the MsgMenu application. The AMSG cell must immediately follow the RMSG cell.

Cell 1 (RMSG)	Plays a prompt to tell the caller how to leave a message, then records the message and stores its message number in the CURRENT MESSAGE buffer.
Cell 2 (AMSG)	Adds the newly recorded message to the message database by storing it in the mailbox identified by the MAILBOX ID buffer. The message will be saved in the message database even if the caller should hang up immediately after recording the message.
Cell 3 (MENU)	Plays a menu that gives the caller three choices, then branches according to the caller's menu choice.

Menu selection 1: review the message

Cell 4 (PMSG)	Plays the message back to the caller so that the caller can review it.
Cell 5 MENU)	Plays another menu to the caller. The caller can return to Cell 1 to rerecord the message, or accept the message as it is.

Menu selection 2: save the message

Cell 9 (PLAY)	Tells the caller that the message has been delivered.
Cell 12 (HANG)	Says good-bye and hangs up.

Menu selection 1: rerecord the message

Cell 8 (DMSG)	Deletes the original message. Next, the application loops back to Cell 1.

Menu selection 2: continue

Cell 9 (PLAY)	Tells the caller that the message has been delivered.
Cell 12 (HANG)	Says good-bye and hangs up.

Menu selection 3: delete the message.

Cell 16 (DMSG)	Erases the message.
Cell 14 (MENU)	Plays a menu. The caller can return to Cell 1 to rerecord or continue.
Cell 12 (HANG)	Says good-bye and hangs up.

Figure 6-4: LeaveMsg page 1

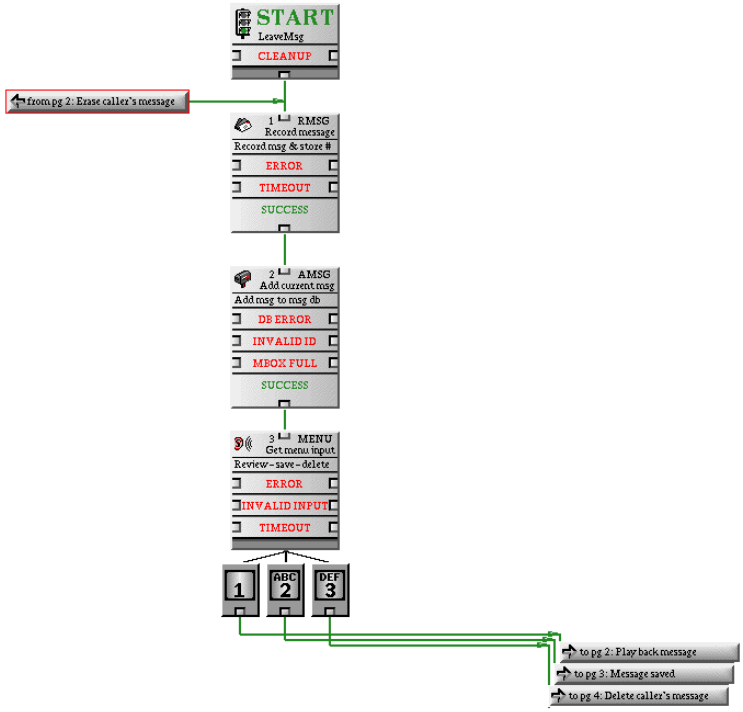


Figure 6-5: LeaveMsg, page 2

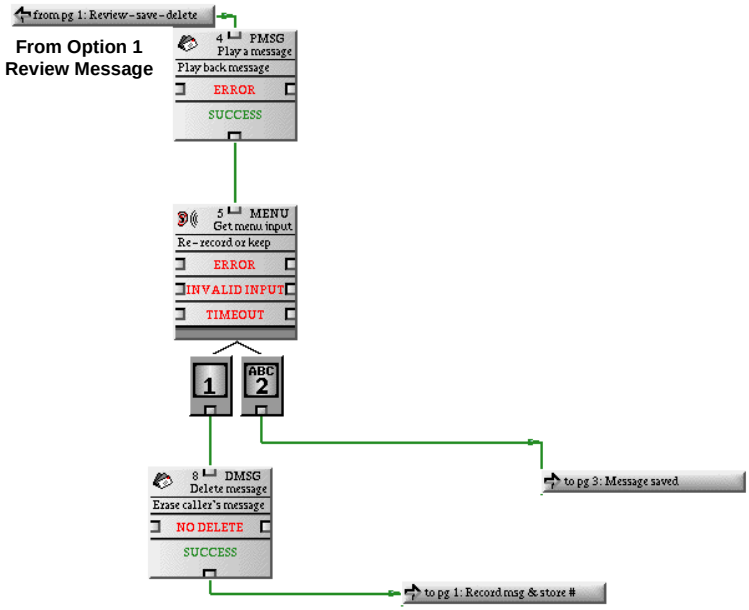


Figure 6-6: LeaveMsg, page 3

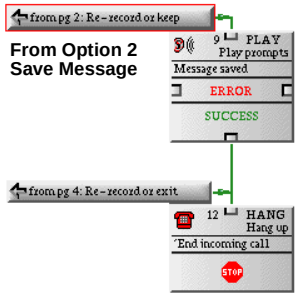
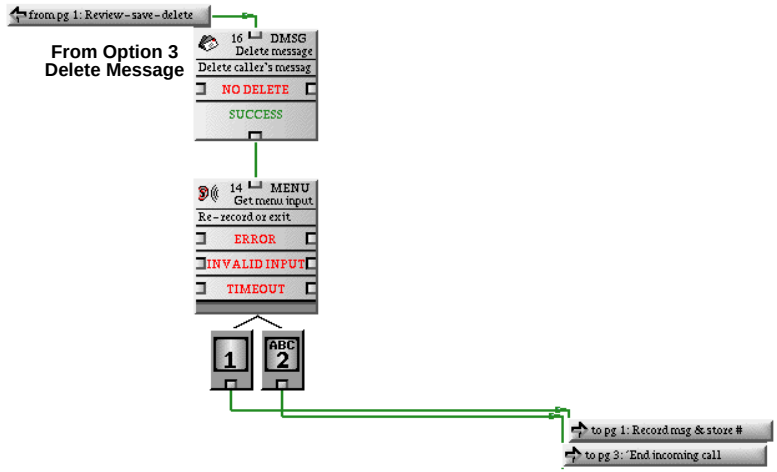


Figure 6-7: LeaveMsg, page 4



RetrievMsg application

Figures 6-8, 6-9, 6-10, and Figure 6-11 show the RetrievMsg application, which allows the caller to retrieve messages from a mailbox one at a time. RetrievMsg begins when the MsgMenu application turns over control of the call with an EXEC cell. The mailbox ID is the DID digits that were received by the MsgMenu application.

Cell 1(GDAT)	Asks the caller to enter a password, then accepts and stores the password
Cell 2 (MSGS)	First, Cell 2 checks to see if the password is valid. If the password is valid Cell 2 accesses the mailbox identified by the mailbox ID, then counts and stores the number of messages that it finds in the mailbox. Next, the application branches to Cell 3. ***Note:*** If there are no messages in the mailbox, then the application takes the “NO MESSAGES” branch.

If the password is invalid:

Cell 10 (PLAY)	Tells the caller that the password was invalid.
Cell 12 (HANG)	Says thank you, good-bye, and hangs up.

If the password is valid:

Cell 3 (PDAT)	Tells the caller how many messages are in the mailbox.
Cell 4 (GMSG)	Tries to retrieve one message from the mailbox. If a message was retrieved, Cell 5 comes next; if there are no more messages, Cell 11 is next.
Cell 5 (PMSG)	If a message was retrieved from the mailbox, Cell 5 plays it to the caller.
Cell 6 (MENU)	Plays a menu that lets the caller choose whether to keep the message that was just retrieved, play it back again, or delete it.

Menu selection 1: review the message

Cell 5 (PMSG)	The application returns to Cell 5 and continues.

Menu selection 2: save the message

Cell 7 (AMSG)	The application saves the message then returns to Cell 4 (get another message).

Menu selection 3: delete the message.

Cell 8 (DMSG)	Deletes the message.
Cell 9 (PLAY)	Tells the caller that the message has been deleted. The application then returns to Cell 4.
Cell 11(PLAY)	Tells the caller that there are no more messages.
Cell 12 (HANG)	Says thank you, good-bye, and hangs up.

Figure 6-8: RetrievMsg, part 1

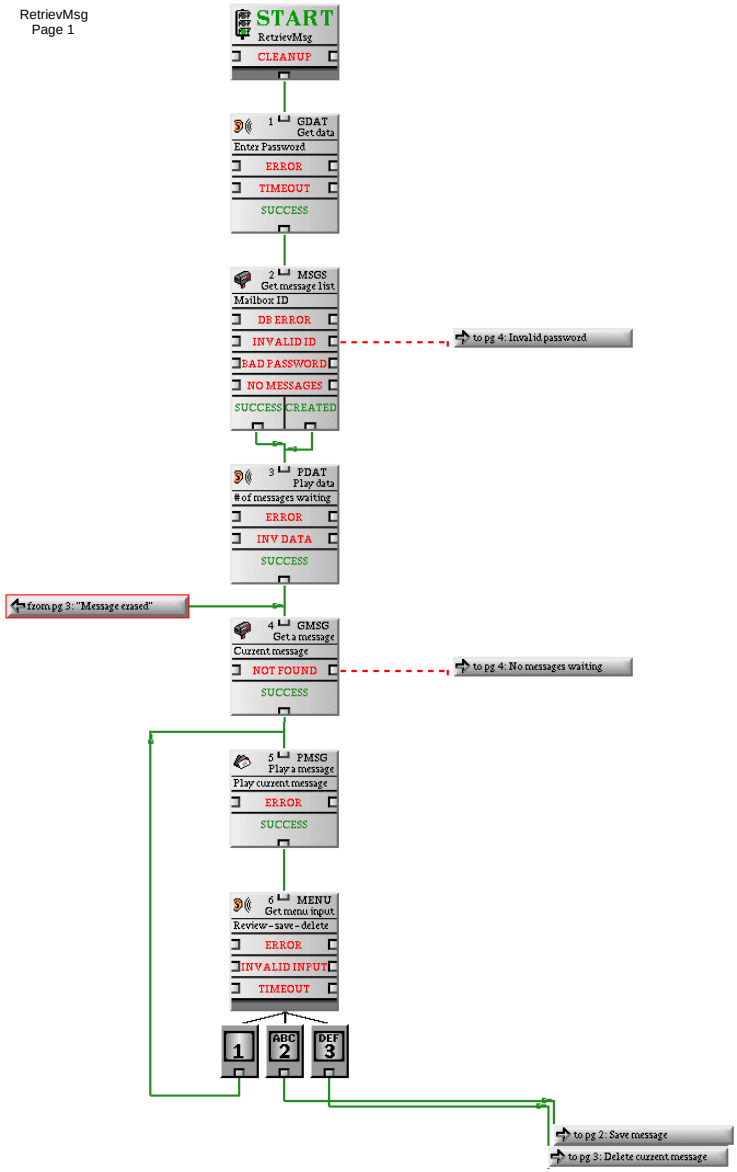


Figure 6-9: RetrievMsg, page 2

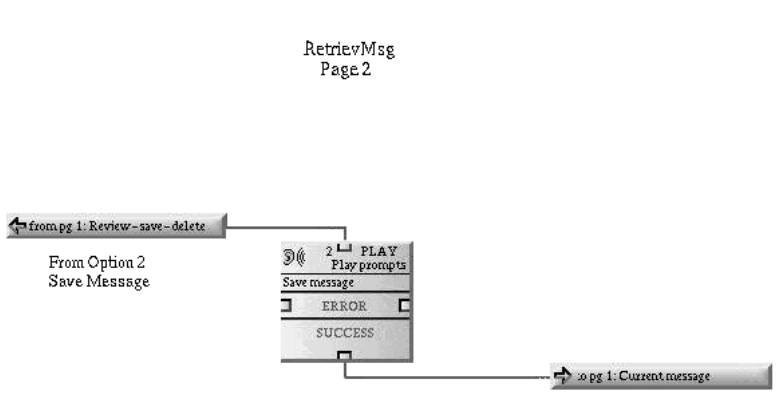


Figure 6-10: RetrievMsg, page 3

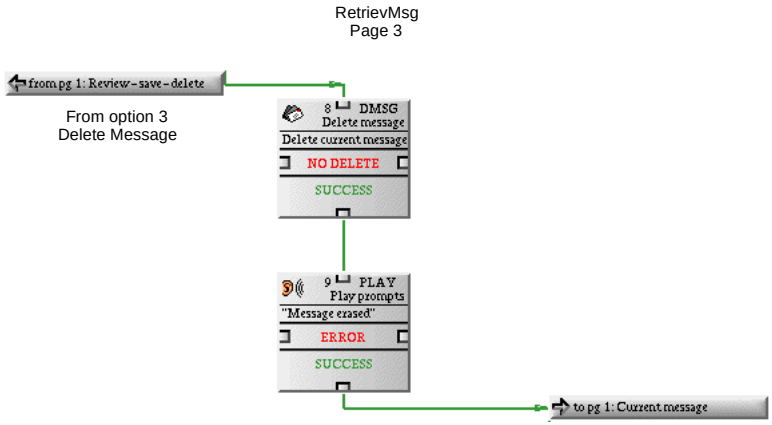
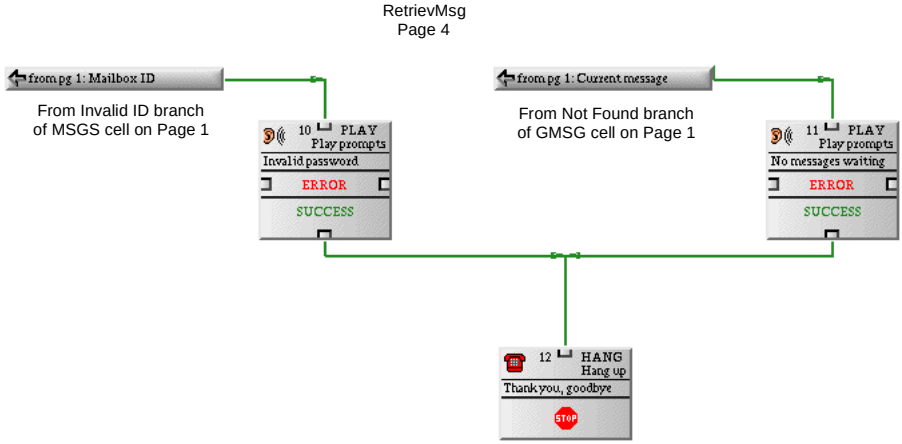


Figure 6-11: RetrievMsg, page 4



Chapter 7: Using information databases

The database facility in AP software is useful in building applications. The AP software information database contains data, in the form of character strings, that can be used as part of the application. Information databases are used to store data.

After you create your database and define the template, you can create, update, insert, or delete existing records from the database. You can perform these functions in one of the following ways:

- Through the application

  — In the example discussed later in this chapter, you can use the application to play the various greetings on specified dates.

  — In addition, you can create applications to check, update, insert new records, and delete existing records from the database using the following cells: CHEK (Check Database), UPDT (Update Information Database), DINS (Insert Records into an Information Database), and DDEL (Delete Records from an Information Database).

- Through the Information Database Editor

  — You can return to the Information Database Editor and read the information from the screen before the database is interfaced to an application.

Figure 7-1 shows an example of using an information database through an application to transfer to an application by capturing and referencing the called number (DNIS). Refer to the section, “Using database information through the application” on page 7-20 for specific information.

Figure 7-1: Example of Using an information database

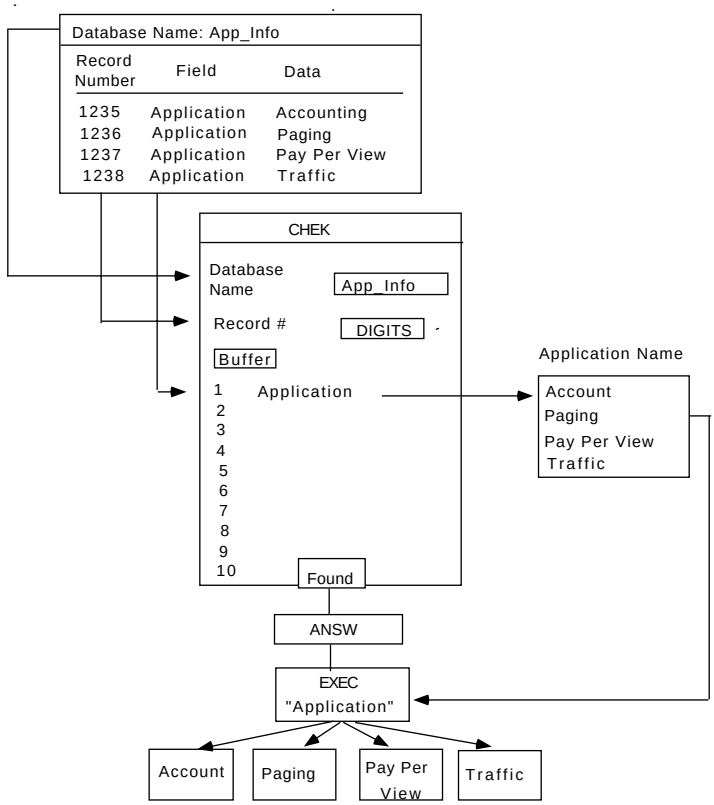


Figure 7-1 illustrates one way to use an information database. The name of the database in this example is App_Info. The record numbers in the database are 1235, 1236, 1237, 1238, and so on. The field descriptor is "Application." The data includes the names of the following applications: Accounting, Paging, Pay Per View, and Traffic.

The database name is stored in the Database Name field of the CHEK cell. The record numbers are stored in the DIGITS buffer in the "*Record #*" field. Note that the DNIS digits are received before the call is answered. AP software can receive DNIS and ANI digits from the MRS and place these digits into the DIGITS and ANI DIGITS buffers of the CHEK cell.

The DIGITS buffer is passed as an input buffer into the CHEK cell, which matches one of the record numbers in the database (for example, 1235). The application name (in this example, Accounting) that corresponds to the record number (for example, 1235) is passed back as output in a buffer (for example, DATA EXCHANGE #1). The buffer (such as DATA EXCHANGE #1, which is the application name in this example) is then passed as input to the EXEC cell. Thus, the information in the buffer DATA EXCHANGE #1 determines which application is executed.

Branching is determined by the data (for instance, Accounting) extracted by the CHEK cell. Therefore, if the CHEK cell retrieves data, such as Accounting, and passes this as input to the EXEC cell, then the application Accounting is executed. If, for example, the CHEK cell retrieves the data Paging and passes this as input to the EXEC cell, then the application Paging is executed, and so on. Keep in mind that the data in DATA EXCHANGE #1 in this example is the application name.

The example in Figure 7-1 shows how an application can be used to branch to multiple applications depending on the DNIS number called and received from the switch.

The information that follows shows another way to use an information database through an example called “Greeting.”

Planning an information database

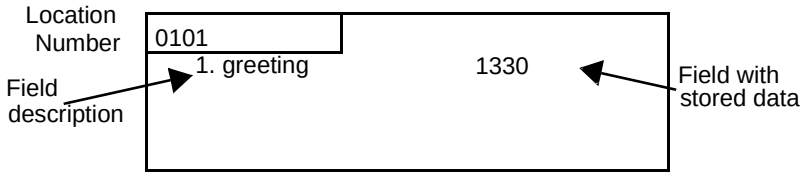
Before you begin to create an information database, read this section for the following topics:

- Understanding record layouts
- Determining information for database records

Understanding record layouts

Each information database consists of a set of records. Figure 7-2 illustrates a single storage location, 0101.

Figure 7-2: Example of a storage location



Note the following activities that take place:

- For each storage location, a unique identification number is assigned.
- You assign descriptors for the template and provide data for fields in the records. For example, the data in the field *Greeting* in Figure 7-2 is the associated prompt number (1330). In the example, the entry in the data field determines which prompt (that is, greeting) to play on the specified date.
- The date is associated with a storage number in the CHEK cell.
- The application name, “Greeting,” is stored in the buffer of the CHEK cell.
- The table of greetings is stored by prompt numbers in the prompts buffer of the PLAY cell.

Information about using the cells is provided later in this chapter.

You can store up to 10 data fields in a single storage location. Each descriptor and data field can contain a maximum of 31 characters. The example used in this chapter shows you how to use stored data to play greetings for holidays and advertising specials on specific dates. In Figure 7-2, the greeting “Happy New Year” is stored as prompt number 1330.

Figure 7-3:

Field descriptions assist in tracking stored data. Each storage-location number in a particular information database contains the same set of field descriptions. You define the field descriptions in the database template. The procedures for creating a template and entering field data are described later in this chapter.

The storage-number locations in the example information database "Greeting" are the month and day, in the format *mmdd*, when each greeting is played to callers. Each record contains one descriptor (that is, greeting) and an associated data field (the prompt number associated with the greeting). Table 7-1 shows the storage location (record number), the date when the greeting is played, and the text of the greeting, followed by the prompt number.

Table 7-1: Storage location

Storage location	Date	Greeting	Prompt number
0101	01011996	Happy New Year	1330
1031	10311996	Happy Halloween	1331
1115	11151996	Happy Holidays	1332
0815	08151996	Fall Specials	1400
1101	11011996	Winter Specials	1401

Storage-location numbers point to the stored data; the storage-location number is not part of the data field. The number of digits in the storage-location number is determined by specifying a number of significant digits when you create the database template, which is discussed later in this chapter.

Determining information for database records

Before you can build an information database, you need to answer the following questions:

- What information do you want to store in the information database?

  You can store a maximum of 10 field descriptions for each storage location; each descriptor and associated data field can contain a maximum of 31 characters (that is, you can store a maximum of 31 characters in the descriptor field and in the data field).

- What is the number of significant digits for your storage-location numbers?

  You can assign either 4 or 5 as the number of significant digits.

- Do you want your application to access data from an information database or a messaging database?

- Do you plan to use the information database in conjunction with an application that also uses a message database? If so, how do you plan to coordinate the information database with the message database?

  To coordinate the two databases, assign one storage-location number for each mailbox and assign the same storage-location number for the mailbox ID number. However, the mailbox number must be created by an application.

For further information, refer to Release 4.0 of the *Symposium OPEN IVR System Administration Guide*, or consult your system administrator.

> ***Note:*** While you can create as many information databases as you choose, AP software allows you to access a maximum of 20 information databases among all applications online.

- What is the name of the database? The name can contain a maximum of eight keyboard characters. However, you cannot use blank spaces. The database in the example in this chapter is named “Greeting.”

After you plan, create, and define a template for your database, you can create, delete, read, or update records contained in an information database. When the information is read, it is stored in buffers that can be referenced by other cells within the application.

Creating an information database

After you plan your database, you can create a database using the Information Database Editor interface. This section discusses each step in the process.

The steps for creating a database include the following:

1 Opening the Information Database Editor.

2 Naming the database.

3 Specifying the number of significant digits.

4 Choosing the database type.

5 Creating the database template (optional step).

6 Creating records.

7 Editing records.

8 Create storage-location numbers (optional step).

9 Add data in each field that you create.

10 Save the information.

11 Exit from the Information Database Editor.

Note: If you prefer, you can enable your application to create storage-location numbers after you create the database. The example in this section, "Greeting," uses the Information Database Editor to create the template and storage-location numbers.

Opening the Information Database Editor

You must run AP software in order to access the Information Database Editor.

To access the Information Database Editor tool follow these steps.

1 Click MB1 on the Information Database Editor button on the Control Panel.

2 From the menu, select Information Database Editor, as shown below.



Alternatively, you can start the Information Database Editor from the system shell by changing the current directory to `exe` in the base directory of AP software.

3 In the `exe` directory, type the following command at the shell prompt and press **<**Enter**>**:

```
sde_start
```

Whether you access the Information Database Editor from the Control Panel or from the system shell, the SELECT A DATABASE screen in the Information Database Editor is displayed.

Figure 7-4 shows an example of a SELECT A DATABASE screen.

Figure 7-4: Select a database screen



Note: Although you can run only one Information Database Editor at a time, you can run multiple applications that simultaneously access information databases.

Naming the database

From the SELECT A DATABASE screen in the Information Database Editor, you can create and name a new database. You can also create a database from the UNIX command line; this procedure is discussed later in this chapter. To name the database, follow these steps.

1 From the SELECT A DATABASE screen in the Information Database Editor, press **<**F3**>** for "New Database."
The following prompt is displayed:

```
Enter The Name Of The Database:
```

2 At the prompt, enter a name for the new database, then press **<**Enter**>**.

The name of the database cannot exceed eight keyboard characters; you cannot use blank spaces in the name.

Specifying the number of significant digits

The number of significant digits is the number of digits in the storage-location number (that is, record number) or range of numbers. Record numbers are often associated with calendar dates or telephone numbers.

1 Name the database and press **<**Enter**>**.
The following prompt is displayed:

```
Enter the # of Digits (between 4 and 5,
inclusive):
```

If you specify 4 as the significant number, the range of record numbers that can be accessed is from 0000 to 9999. If you specify 5, the range of record numbers is from 00000 to 99999.

2 Press **<**Enter**>** to accept the default value (4), or type 5 and press **<**Enter**>** to specify 5 as the significant number of digits.

Note: Once you specify the significant number of digits, the number cannot be changed unless you delete and recreate the database. Furthermore, you cannot export a database with a higher number of significant digits (5) to a database that uses fewer significant digits (4).

Choosing the database type

After you specify the number of significant digits, you are prompted by the Information Database Editor to specify a database type.To choose a database type, follow these steps.

1 At the following prompt, enter **i** to choose Information Database:

```
Enter the Database Type (Info or Message): i
```

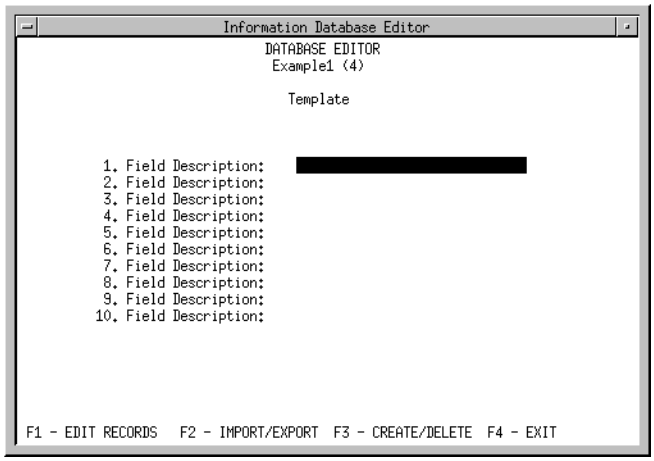
Refer to Chapter 6 for information about message databases.

2 Press **<**Enter**>**.

The Template screen is displayed.

An example of a template screen for an information database called “Example1” is illustrated in Figure 7-5.

Figure 7-5: Template screen



Database templates

After you create and name the database, determine the significant number of digits for the storage-location numbers, and specify the information database type, then you create the database template (if you choose to use a template). The template consists of field descriptions (for example, Greeting). The field descriptions that you define in your template are used for every record in the database.

After you specify the database type and press <Enter>, the Template screen is displayed. In the example presented in this chapter, the name of the database is “Greeting.” The number of significant digits for the database record numbers is displayed beside the database name. The number of significant digits in the “Greeting” database is four.

From the Template screen, you can perform the following functions:

- Enter field descriptions in the template.
- Press <F1> to access existing records.
- Press <F2> to import and export databases.
- Press <F3> to create or delete records.

- Press <F4> to exit from the Template screen and return to the SELECT A DATABASE screen.

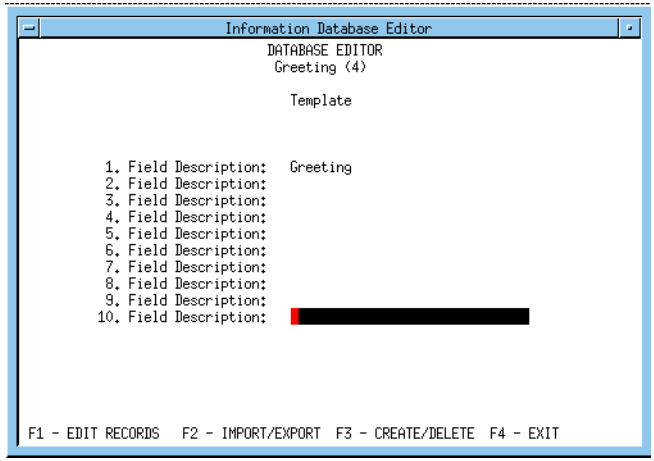
Creating the database template

To create a template, which is composed of field descriptions that are used by each storage-location number in the database, follow these steps.

1 Enter a description next to the related field. Press **<**Enter**>** or press the Down Arrow key to move the cursor to the next field.

In the “Greeting” database, the data in the field description is the associated prompt number that is played to callers on the specified date. As described later in the chapter, when associated with an application, the CHEK cell reads the data from the field into an output buffer. This output buffer is passed in as an input prompt number to the PLAY cell; for example, the message “Happy New Year” is played to the caller. Note that the greetings themselves, such as “Happy New Year,” are stored as prompt numbers in a buffer contained in the PLAY cell.

Field descriptors can include a maximum of 31 keyboard characters, with the exception of commas. Press <Backspace> to delete characters. Blank spaces at the beginning or end of a field description entry are deleted, as shown below.



2 Repeat this procedure to enter field descriptions for each field. The maximum number of fields in a record is 10.

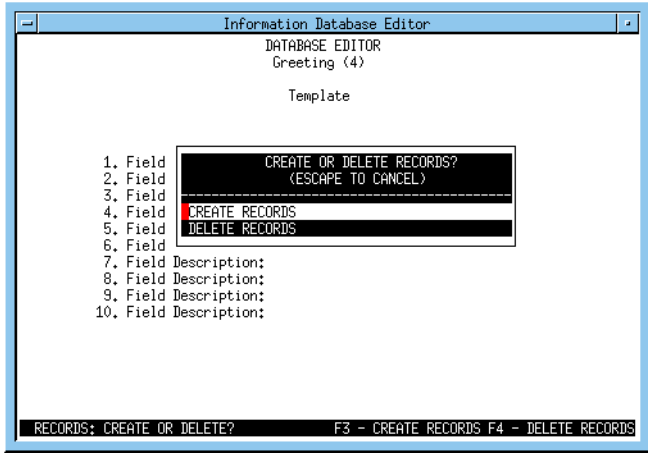
After you enter the field descriptions, you can create records.

Creating records

To create records from the Template screen, follow these steps.

1 From the Template screen, press **<**F3**>** to create a record or a range of records.

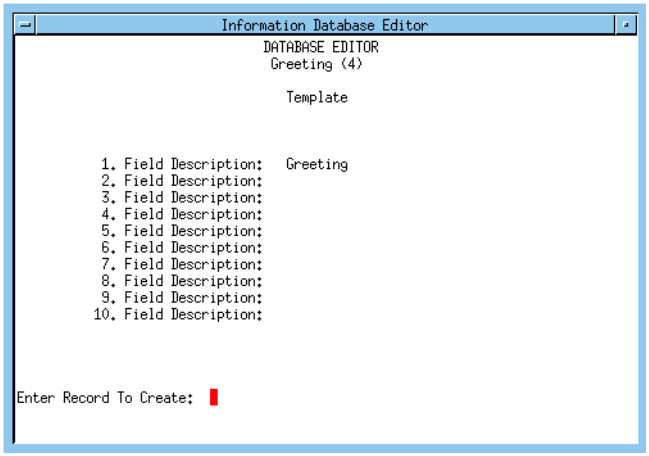
The CREATE OR DELETE RECORDS window is displayed.



2 Using the cursor keys, highlight “CREATE RECORDS” and press Enter.

3 You are prompted to enter a record number or range of record numbers.

.



4 Enter either one record number or a range of record numbers, then press **<**Enter**>**.

5 To enter a range, type the beginning number, a hyphen, and an ending number without blank spaces. For example, to create a range of numbers from 1 to 4000, type the following information without spaces:

1-4000

In this example, the range of record numbers is from 0001 to 4000; the number of significant digits is 4.

6 Repeat these steps to create additional records.

7 After you enter the number or range of numbers and press **<**Enter**>**, the Template screen is displayed.

Editing records

To edit these records, follow these steps.

1. Press **<**F1**>** from the Template screen to edit the record.
   If you created one record number, that record is displayed. If you created a range of record numbers (for example, 8 — 10), the first record in the range (that is, 8) is displayed.
2. Add data in the record fields.
3. After you finish entering data in the record or records, press **<**F4**>**.

   The "Update Database" window is displayed.
4. To save your changes, select YES and press **<**Enter**>**.

After you create the record or a range of records, you can save your changes and exit from the Information Database Editor. (See "Exiting from the Information Database Editor" on page 7-19 for procedural information.)

Editing the template

You can edit the database template and change field descriptions. To do so, follow these steps.

1. Select a database, and press <Enter>. From the "Record Number" screen, press **<**F1**>** for "Edit Template." The Template screen is displayed.
2. Using the arrow keys, move the cursor to the field you want to change, then enter the new or revised description in the appropriate field.
3. Press **<**F4**>** to save your changes and exit from the template.

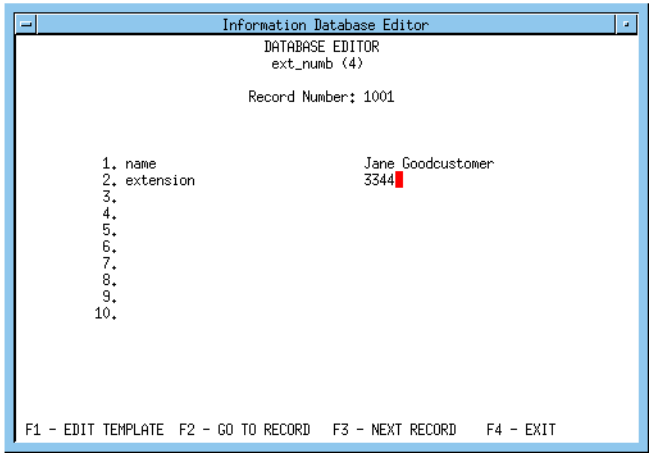
Inserting or changing data in records

Once you create the records, you can add or change the information in them. You can edit the data at any time, even when the database is being used by applications.

To edit the current data, follow these steps:

1 From the Template screen, press **<**F1**>** for "Edit Records" to add or change existing information in a record.

The "Record Number" screen is displayed. The lowest existing record number is displayed (in this example, Record Number 1001).



2 You can either enter data in the record that is displayed or you can access another record.

3 Use the arrow keys to move the cursor among the data fields in the record. Edit the information as appropriate.

4 If you type over existing information, the current data is deleted. If you type incorrect information, press **<**Backspace**>** to delete characters.

5 After you make your changes, press **<**F4**>** to exit from the record.

Accessing record numbers

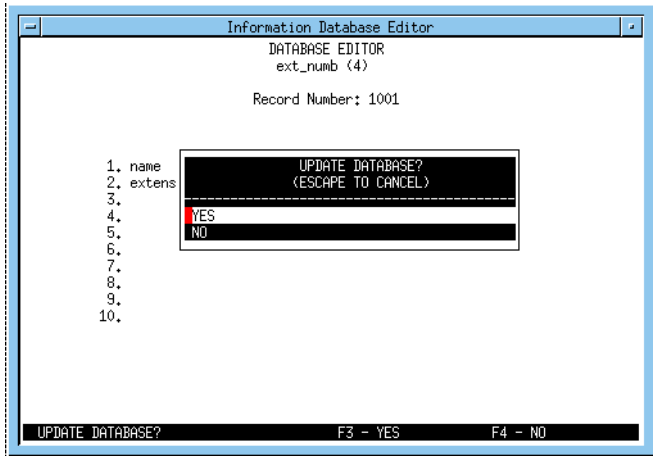
The following procedures explain how to access a record number other than the number displayed on the Record Number screen. The current record number is shown on the third line.

You can access either the next consecutive record number or another record number.

To access the next consecutive record number, follow these steps.

1 From the Record Number screen, press **<**F3**>** for “Next Record.”

If you edit data in the current record, the “Update Database” menu is displayed, which enables you to update the database with the information in the edited record.u



2 To save your changes in the database, highlight YES. If you do not want to save your changes, highlight NO.
Press **<**Enter**>**.
The next consecutive record number is displayed. Note that if the last record number is displayed and you press **<**Enter**>**, the first (that is, the lowest) record number is displayed next.

Accessing a specific record number

To access a specific record number in the console window, follow these steps.

1 From the Record Number screen, press **<**F2**>** for “Go To Record.”

The following prompt is displayed:

```
Enter Record:
```

2 Enter the number of a single record or a range for a range of record numbers, and press **<**Enter**>**.

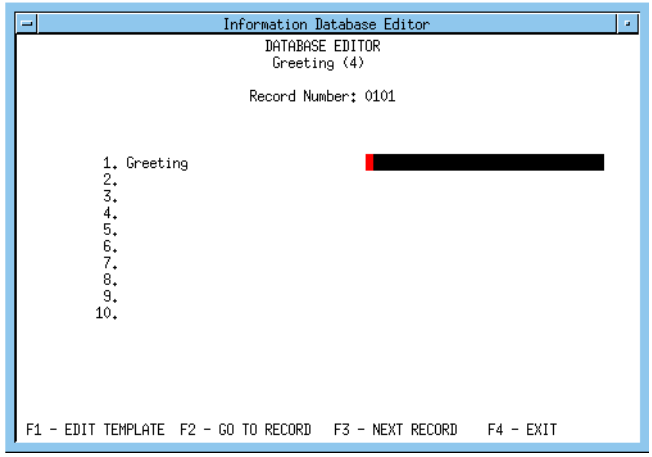
The record number is displayed near the top of the screen. If you specify a range of numbers, the first record number in the range is displayed, followed by the next sequential record number, and so on.

3 If you changed the data in the record at the current location, the “Update Database” menu is displayed when you exit from the record. To save your changes, use the arrow keys to highlight YES; if you do not want to save your changes, highlight NO. Press **<**Enter**>**.

The “Select a Database” screen is displayed.

The following shows the screen that is displayed when you access record 0101.

.



Deleting record numbers

When you delete a record number, the data that is stored in that record is deleted, follow these steps. To delete a record number from the information database,

1 From the Template screen, press **<**F4**>** for “Delete Record.”

The following prompt is displayed:

```
Enter Record to Delete:
```

2 Enter the number of a single record, or enter a range of record numbers that you want to delete.

3 Press **<**Enter**>**.
The record number or range of numbers is deleted.

Exiting from the Information Database Editor

After you complete your work, follow these steps. To exit from the Information Database Editor,

1 From the Record Number screen, press **<**F4**>** for “Exit.”
If you edited data during the session, the “Update Database” menu is displayed.

2 To save your changes, use the cursor to highlight YES; if you do not want to save your changes, highlight NO.

3 Press **<**Enter**>**.
The “Select a Database” menu is displayed.

4 Press **<**F4**>** to exit.
The Information Database Editor closes.

Loading an existing information database

Loading a database is the process of reopening a database. You can access an existing database at any time, even when the database is currently in use by an application, in order to add, edit, or browse information.

To load an existing database and add, edit, or browse information, follow these steps.

1 Open the Information Database Editor from the Control Panel.

2 From the "Select a Database" screen in the Information Database Editor, use the arrow keys to highlight the name of the database you want to load, then press **<**Enter**>**.

3 Once you load an existing database, you can perform the following functions:

- Create record numbers.
- Add or edit current data in the record fields.
- Edit the database template.
- Delete record numbers.

4 Exit from the Information Database Editor.

Using database information through the application

This section describes how you can use an information database through an application to do the following:

- Play back specialized greetings.
- Maintain the information database by checking existing records, as well as creating, inserting, updating, and deleting records.

Playing back specialized greetings

The "Greeting" database example describes one way to use an information database. The record numbers represent the date, in *mmdd* format, when the greeting is played: 0101 (January 01), 1031 (October 31), 1115 (November 15), 0815 (August 15), and 1101 (November 01).

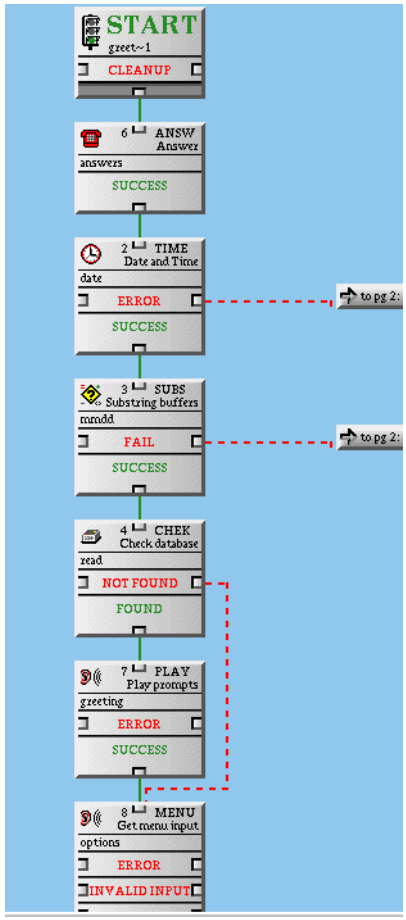
Each record contains one data field "Greeting," which is represented by the associated prompt number specified in the prompts buffer of the PLAY cell. Refer to Table 7-1 to see the text and associated prompt number for each greeting.

The application used by the information database "Greeting" consists of the following cells:

- ANSW: answers the phone call on the current channel.
- TIME: updates the buffers with the current date in the format *mmddyyyy*. The TIME cell passes the current date to the SUBS cell in the DATE buffer.
- SUBS: extracts the first four digits of the date (that is, *mmdd*), which is passed to it by the TIME cell. This format, *mmdd*, is then passed to the CHEK cell as an input record number.
- CHEK: looks up the record number and reads the data in the fields of the records into a maximum of 10 buffers. (In the example in this chapter, the data from the data field in each record is read into one buffer. The content of this buffer is a prompt number.)
- PLAY: plays the specific greeting associated with the prompt number in the prompts buffer if the current date in the CHEK cell matches the record number. If the current date does not match the record number, then the CHEK cell exits through the "not found" exception branch and bypasses the PLAY cell.

Figure 7-5 illustrates the call flow of the cells for the application that plays the various greetings on the specified dates. Refer to Release 4.0 of the *Symposium OPEN IVR Cell Catalog* for additional information about each cell.

Figure 7-6: Call flow



Locating and extracting data from an information database

This section describes how you create applications using the following cell types to find and extract data from an information database in order to make the data available to other cells in the application.

CHEK cell

The CHEK cell contains the following parameters:

- Database Name

  Name of the database that is checked.

- Record Number

  System or user-defined buffer that contains the record number to be checked.

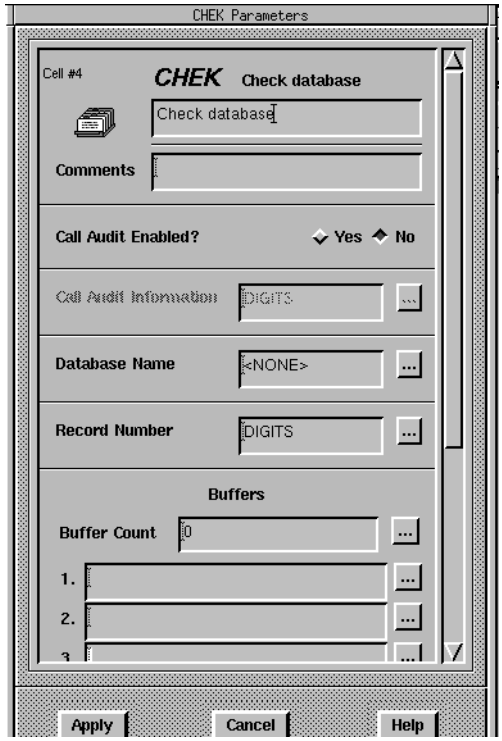
- Buffer Count

  Number of output buffers that are used to copy data from the specified record (maximum = 10 buffers).

  Note: A one-to-one match exists between the fields in the record and the buffer fields in the CHEK cell. You cannot skip record fields or buffers. If a match to the record number exists, then all the fields in the record are copied to the specified buffer. Once the data is copied into buffers, then any cell in the application can use that data.

In the CHEK cell, identify the name of the information database you want the application to access and the names of the buffers in which to place the extracted information. Note that the CHEK cell can check only one information database at a time. Figure 7-7 shows the parameters in the CHEK cell.

Figure 7-7: CHEK cell



The CHEK cell can be used both to determine whether a record exists for a specified record number within the selected information database and to retrieve data. If the record does not exist, the CHEK function branches to the “not found” next cell. If the CHEK cell successfully locates the specified record, then information from the record is copied into buffers specified in the CHEK cell. These buffers are now available for use by other cells within the application. Data from all the fields in the record are read.

The process of updating an existing database record involves the following steps: reading the record (CHEK cell), changing the information contained in one or more buffers (GDAT cell), and then writing the information back into the database (UPDT cell). The information that follows explains how to update an information database using cells in an application.

- GDAT cell: places updated information into an appropriate buffer. Note that you need a separate GDAT cell for each buffer (that is, data field) that you update.
- UPDT cell: writes updated data back to the information database by inserting the buffers of data, which includes the updated data from the GDAT cell, into all fields of the record.

Use the following cells to insert or delete a record number:

- DINS cell: inserts the record number that is specified in the buffer into an information database. Note that only the record number is inserted.

  It is important to include the DINS cell in your application so that you can insert additional records into the database if you need to insert records through your application.
- DDEL cell: deletes a record from an information database.

UPDT cell

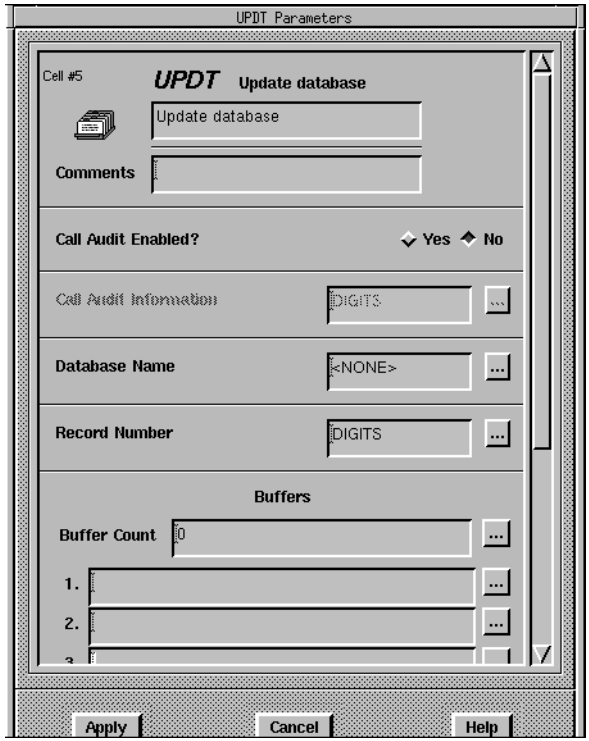
Updating an existing database record involves reading the record (CHEK cell), changing the information contained in one or more buffers of the GDAT cell, and then writing the information back into the database through the UPDT cell. The UPDT cell enables the application to update records in an information database. You can use the UPDT cell to write to a new record or to write to an existing record that contains stored data.

> ***Note:*** Simultaneous reading and writing to an information-database record can result in the corruption of data stored in the record.

To update the fields in a record, information is passed into the record that you specify using either system-defined or user-defined buffers. If the specified record number is found in the information database, as many as 10 buffers of information can be inserted into the record. Refer to Figure 7-8 for an illustration of the UPDT cell.

Note: You must create the record before you can use the UPDT cell. In addition, you must list each input buffer sequentially (that is, you cannot skip buffers). For instance, you must list buffers in the following order: 1, 2, and 3. You cannot skip from buffer 1 to buffer 3.

Figure 7-8: UPDT cell



The UPDT cell includes following parameters:

- Database Name

  Name of the database that is checked and then updated by the UPDT function.

- Record Number

  Number of the record to update in the database.

- Buffer Count

  Number of system or user-defined input buffers that are used to copy data to the database record.

  Note: Each buffer cannot contain more than 31 characters.

The following example shows how you can use the UPDT cell in conjunction with the “Greeting” information database. The example illustrates one of the five records.

The values for the parameters for the UPDT cell in this example are as follows:

Database Name	Greeting
Record Number	0101
Buffer Count	1

The UPDT cell uses one input buffer. In this example, the buffer “Greeting” will contain the following information:

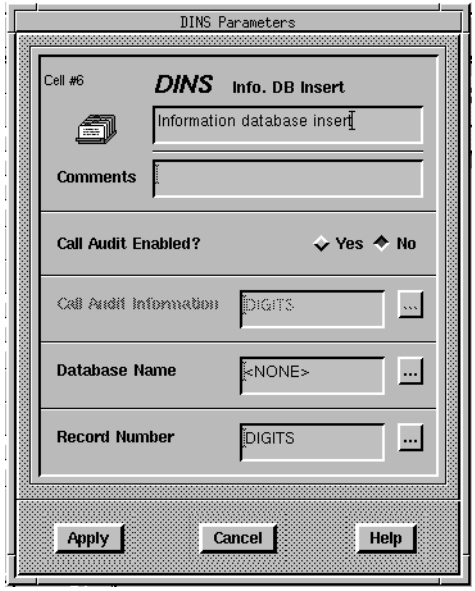
Greeting 1330

When used in an application, the UPDT cell looks for the “Greeting” information database and locates record number 0101. When record 0101 is located, the information is copied from the buffer and inserted into the record.

DINS cell

The DINS cell enables you to insert an additional record number into an information database. However, if the record number that you specify already exists, the application takes the “Record Exists” branch of the DINS cell, and a new empty record is not created. Figure 7-9 illustrates the parameters in the DINS cell.

Figure 7-9: DINS cell



The information that follows shows you how to insert a new record number into a database.

The parameter values for the DINS cell are as follows:

- Database Name

  Name of the database where the record number is inserted.

- Record Number

  System or user-defined buffer that contains the record number that is inserted into the information database.

  Note: The record number is the only information inserted into the database.

For example, assume that you use the DINS cell to insert a new record number, 0315, into the “Greeting” information database. The values for the parameters for the “Greeting” database are as follows:

Database Name	Greeting
Record Number	0315

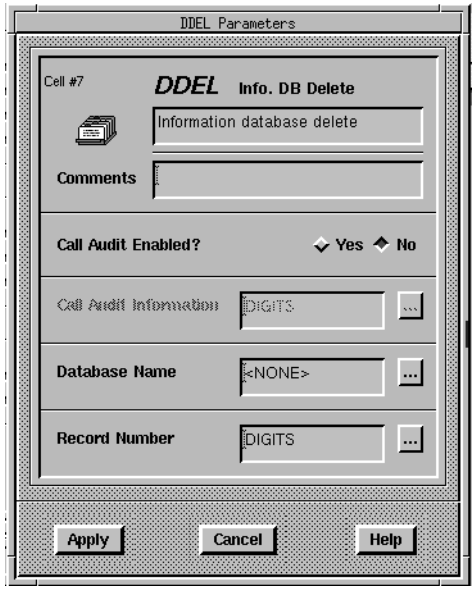
If the DINS cell is successful and record number 0315 does not currently exist, then a new empty record (0315) is inserted into the database.

DDEL cell

The DDEL cell enables you to delete record numbers from an information database. To delete a record, you must specify its number. Figure 7-10 shows the parameters in the DDEL cell.

Note: You cannot recover a record once it is deleted.

Figure 7-10: DDEL cell



The parameters for the DDEL cell include the following:

- Database Name

  Name of the database where the record number is currently located.

- Record Number

  System or user-defined buffer that contains the record number you choose to delete.

  Note: If the specified record contains VSNs that are not included in the system default prompts (for example, if you assign a prompt to record number 0315, such as "Happy St. Patrick's Day"), then the VSN data and number are not automatically deleted when you delete the record number. You should delete the VSNs before you delete the record number. Otherwise, those VSNs are not available for use once you delete the record.

Formats for exporting and importing data

From the Template screen in the Information Database Editor, press <F2> to export or import information databases. The export function writes the contents of the information database to a specified ASCII file in the `sys_files` directory.

AP software supports two formats for exporting data: internal and CSV (comma-separated value). The internal format is compatible with AP software local databases. The CSV format enables you to do the following:

- Export AP software local-database data to other software packages.
- Import data from other software packages.

The internal format for the export file is as follows:

```
record.field:data
```

where

- *record* is the record number in the database where the data is located.
- *field* is the field number in the record where the data is located (numbers 1 through 10).
- *data* is the information entered into the database by users.

If the same data resides in consecutive records or fields, then you can specify the record and field numbers as ranges; separate the range of numbers with a hyphen.

The CSV format of the export file is as follows:

```
record#, field1,field2,field3,...,field10
```

where

- *record#* is the entry in the database where the data is located.
- *field1 — field10* is the data in these fields.

Exporting data

To export an information file, follow these steps.

1 Open the Information Database Editor.

2 Select the database that you want to export, then press **<**Enter**>**.

3 From the Template screen, press **<**F2**>** for "IMPORT/EXPORT."

The "IMPORT OR EXPORT INFO" menu is displayed.



4 Use the arrow keys to highlight "EXPORT INFO", then press **<**Return**>**.

The following prompt is displayed:

```
Enter format (CSV | internal):
```

Note: You can choose only one format. See the section on "Formats for exporting and importing data" on page 7-30 for additional information.

5 After you specify the format, the following prompt is displayed:

```
Enter File name:
```

6 Enter the name of the ASCII file for the exported data, then press **<**Return**>**.

The database that is displayed is exported to the ASCII file that you created in the `sys_files` directory.

7 Press **<**F4**>** to exit.

Importing data

Before you import data, create a database for your data to be populated.

To import data, follow these steps.

1 Open the Information Database Editor.

2 Select the database where you want the data imported from the "Select a Database" screen, then press **<**Enter**>**.

The Template screen is displayed.

3 Press **<**F2**>** for "Import/Export." The "IMPORT OR EXPORT" menu is displayed.

4 Use the arrow keys to select "IMPORT INFO", then press **<**Return**>**. The following message is displayed:

```
Enter format (CSV | internal):
```

Note: You can choose only one format. See the section on "Formats for exporting and importing data" on page 7-30 for more information.

5 Select the format and press **<**Enter**>**. The following prompt is displayed:

```
Enter File name:
```

6 Enter the name of the database for the imported data, then press **<**Enter**>**.

Your database is imported from the ASCII file that you created in the `sys_files` directory. The database that is displayed on the screen is populated with the data.

7 Press **<**F4**>** to exit.

Performing database functions from the UNIX command line

You can perform the following database functions from the UNIX command line:

- Create a database.
- Export database information to an ASCII file.
- Import information from an ASCII file to a database.
- View command-line messages.

Creating a database

To create an information database from the command line, enter the command using the following format:

```
$ sde -c -n databasename -d n -t type
```

where

- *sde* specifies the Information Database Editor.
- *-c* creates the database from command-line input. When you select this option, you must also enter the database name, number of significant digits, and type of information database, as follows.
- *-n databasename* names the database (maximum of eight characters with no blank spaces)
- *-d n* sets the number of significant digits for the database (four or five).
- *-t type* defines the type of database: i for information databases; m for message databases.

Example:

To create an information database named “INFOAUTO” with five (5) significant digits, type the following command:

```
$ sde -c -n INFOAUTO -d 5 -t i
```

Exporting database information to an ASCII file

1 To export information from a database to an ASCII file from the UNIX command line, enter the command using the following format:

$ sde -e *filename* -f *format* -n *databasename*

where

- -e *filename* specifies that you are exporting information and designates the name of the target ASCII file.

  Note: If the file name that you specify already exists, you will overwrite the existing file when you execute this command. If the file name that you specify does not exist, a new file is created.

- -f *format* specifies the format for the following options:

  — csv: comma-separated value

  — internal: AP software

- -n *databasename* specifies the name of the database.

Example:

To export database cust_db into the file name customer in csv format, type the following command:

```
$ sde -e customer -f csv -n cust_db
```

Importing information from an ASCII file to a database

To import information from an appropriately formatted ASCII file to a database from the UNIX command line, follow these steps.

1 Enter the import command using the following format:

$ sde -i *filename* -f *format* -n *databasename*

where

- -*i filename* specifies that you are importing information and designates the name of the ASCII file from which the information is imported.

- -f *format* specifies the format for the following options:
  - — csv: comma-separated value
  - — internal: AP software
- -n *databasename* identifies the name of the database to which the information is imported.

  Note: If the name of the database that you specify already exists, you will overwrite the existing name when you execute this command. If the name of the database does not exist, then you can create the name from the command line when you import the information.

2 Specify the import command in the following format:

$ sde -i *filename* -f *format* -n *databasename* -c

Examples:

To import the file customer into database cust_db in internal format, type the following command:

$ sde -i customer -f internal -n cust_db

To import the file customer into a new database, new_db, in csv format, type the following command:

$ sde -i customer -f csv -n new_db -c -t i -d 5

You can create the data using the nedit editor instead of creating the database through the console. By using the nedit editor and the import function, you can create multiple records at the same time; if you use the console, you can create only one record at a time.

Using nedit editor to create multiple records

To create data using the nedit editor, follow these steps.

1 Create the database.

2 Create the data.

3 Save the database.

4 Display the contents of the text file on your screen (that is, the record number and field data) using the `more` or `cat` UNIX command.

5 From the `sys_files` directory, use the nedit editor to create a file that you can import into an existing database, such as `ext_num`.

6 Enter the record data into the file in the appropriate format (for example, the format in the following template is *record number, name, extension number*):

1266, Harry Shaw, 9654

1267, Nadia Klemziak, 8462

1268, Kampeng Vong, 3265

7 Select SAVE to save file `ext_num`.

8 From the template screen in the file `ext_num`, press **<**F2**>** to select the Import function.

9 Select CSV format and specify the new file name, `ext_num`.

Using SDE commands from the UNIX command line

Table 7-2 lists the sde commands and describes their related functions.

1 To display the following sde commands, type the following command at the UNIX prompt and press **<**Return**>**:

`$ sde -`*`function`*

For example: $ sde -create

Table 7-2: SDE commands

Command	Function
-verbose	Sets verbose mode.
-rev	Displays the revision level, then exit.
-branch	Sets the branch.
-create	Creates a database from the command line using the -n (database name) option.
-type	Sets the type of database (i or m). Specify this command when you create a database using the -c option.
-name	Names the database. Specify this command when you create a database using the -c option.
-digits	Specifies the number of significant digits for the record numbers. Specify this command when you create a database using the -c option.
-export	Exports data to an ASCII text file. Specify this command when you name the database using the -n option.
-import	Imports a data file into an information database. Specify this command when you name the database using the -n option.
-format	Specifies the format for the csv and internal options, [csv|internal]; specify either csv or internal. Specify this command when you use the -e (export) or i (information database) options.

Viewing command-line messages

All command-line messages are logged into the Transaction Log. To view these messages, click the Reports icon in the Control Panel and open the Transaction Log.

Chapter 8: Creating User Functions

User Functions, an advanced feature of AP software, allows you to interface your own code to a AP software application in order to perform custom activities not built into AP software.

To use User Functions effectively, you should be familiar with the C and C++ programming languages and the operating system running on the AP software application processor. You should also understand the procedures presented earlier in this guide for designing and building applications.

This chapter covers the following topics:

- Understanding the User Function process
- Designing a User Function
- Coding a User Function
- Building and testing a User Function
- Including a User Function in an application
- The `usr.c` source file template
- Advanced User Function techniques

Note: AP software supports User Function development using the C++ language. You can create C++ User Functions using the C templates supplied with Nortel software. You can also add new C++ code to your existing User Functions, without restriction. Although these C templates do not include C++ specific language features, they will compile under a C++ compiler.

As a User Function developer, you write the code that is executed for each function code. In this executable code, you can use any appropriate development model, either functional decomposition in C or an object-oriented model, and develop your code in C++.

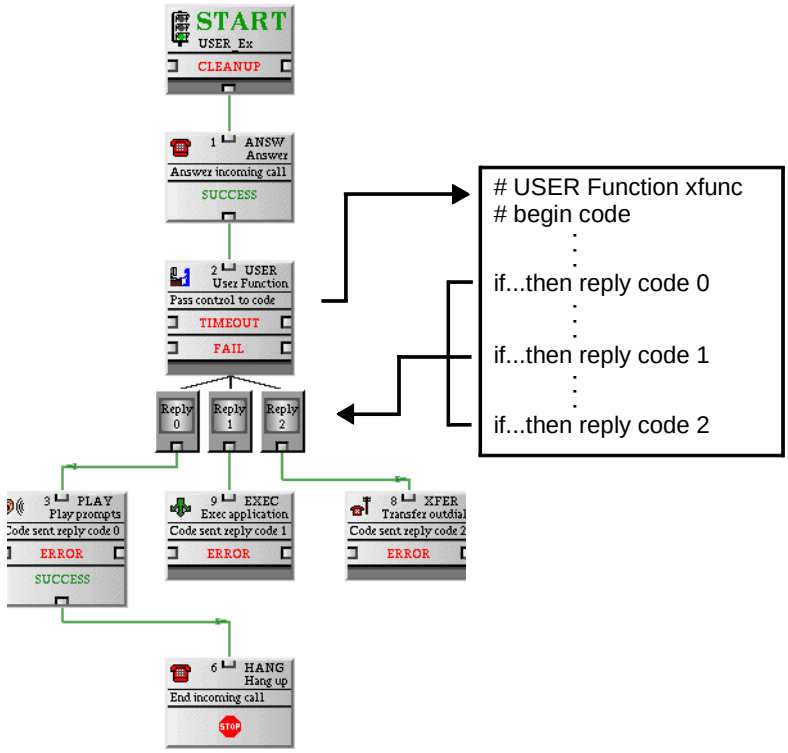
The basic User Function template sets up the appropriate mechanism for communications between OPEN IVR software and the applications that access your User Functions. In addition, the template sets up the basic dispatch mechanism for accessing the desired function codes. This dispatch is currently a C switch statement that is indexed by the numeric function code supplied as input to the User Function request.

The User Function is described further in this chapter.

Understanding the User Function process

Before you code your own User Functions, you must be familiar with how an application interacts with a User Function and how the process that is running (herein called the User Function process, as illustrated in Figure 8-1) is shared with the channels on the system.

Figure 8-1: User Function process



Accessing the User Function from the call flow

AP software accesses your User Function through the USER cell. Each USER cell in your application needs two pieces of information to successfully navigate to and from your User Function: a function code and a reply code.

Function code

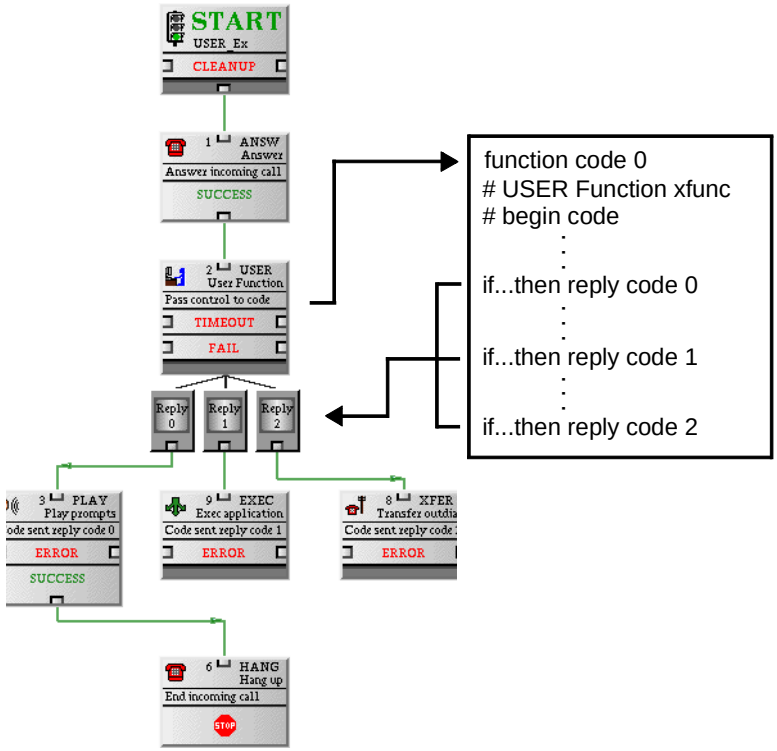
Place the function code value in your User Function at the point where you want a USER cell to begin processing, that is, the access point for the specific CASE statement in your C program (0 — 99).

Reply code

Place the reply code in your User Function at the point where you want the User Function to return control to your AP software application. You can specify between 1 and 10 reply codes to handle different contingencies within the User Function.

Figure 8-2 illustrates an application that passes control to a User Function `xfunc` through the USER cell with function code zero. This example uses three reply codes, each one carrying out a different task.

Figure 8-2: Application passes control to the User Function xfunc



User Function process

When you load and execute an application that references a User Function, the following sequence of events occurs:

1 When your application is loaded, the operating system loads and starts a process for the User Function referenced by the USER cell. A process is generated for each unique User Function name. This process is shared among all channels assigned to the application and does not perform any tasks until a request is received in the message queue. This process is also shared by all applications running in the application processor (AP).

2 Your application is started.

3 When a USER cell in your application is executed, the USER cell passes a message to the corresponding User Function process that includes a set of input buffers and a function code to indicate which of the user-defined tasks to execute. The USER cell then transfers control to the User Function process, and the application on the associated channel is suspended.

4 The User Function process begins executing the specified user-defined function that is identified by function code. Meanwhile, AP software continues processing calls on the other channels. When the User Function process completes processing your task, it passes to the application a set of output buffers if the information is to be returned, and a Reply Code. Then control is returned to the application. The application on the associated channel resumes execution.

5 The application branches to the next cell, which is determined by the Reply Code.

6 When all applications referencing the same User Function are unloaded, the User Function process is aborted.

Input and output buffers

Input and output buffers provide a flexible way to transmit data between the application and the User Function. The application can send the User Function up to 40 input buffers, which can include any AP software system buffers or user-defined buffers. Similarly, the User Function can return up to 40 output buffers to the application. The total number of characters stored in each input buffers to the User Function cannot exceed 1920 characters. The total number of characters passed from the User Function to the application through output buffers cannot exceed 1920 characters per buffer. Data in excess of 1920 characters is truncated.

Reply Code

When the User Function completes its current task, it returns a Reply Code as a value from 0 to 9. When the application resumes, the USER cell checks the Reply Code to determine which cell to execute next. The USER cell has up to 10 branches; it uses the Reply Code to determine which of these branches to take. If you want to force an error code, select a Reply Code and the next cell specifically for errors.

Source file

Included in the AP software is a source file called `usr.c` that you can use as a template for creating User Functions. This template contains the code that you need to initialize and operate a User Function process. Copy this file and insert your own code at a few specified points to create a User Function. Later in this chapter, a full list of the `usr.c` file is presented.

WARNING

Never directly modify the `usr.c` template. First, make a copy. Then, modify your copy. In this way, you can use the template for another User Function.

During execution of your application, your User Function does the following:

- reads the message sent from the USER cell

- determines which user-defined function and specific function code number are selected
- executes the user-defined function, updating output buffers as necessary
- sends a message back to the USER cell with a Reply Code

  Note: The `usr.c` file includes init and halt routines. You can use these routines to insert a startup or exit function in your User Function.

The execution of User Functions can occasionally take a considerable amount of time, depending upon the complexity of the tasks associated with the user-defined function. In order to inform the caller that the application is in progress, the application developer can provide prompts for the caller to listen to while the user-defined function is executing. The prompt or prompt list can include, for example, prerecorded announcements or music. Once the processing of the user-defined function is completed, the application can abort the playing of prompts or allow the current prompt or prompt list to continue playing until it is completed and then resume the call flow in the application.

Error handling

Whenever a user-defined function encounters an error, ensure that the error message is written to the event log. Informational messages can also be sent to the event log. To view the error messages, you can run the Transaction Log Report from the Report tools on the Control Panel.

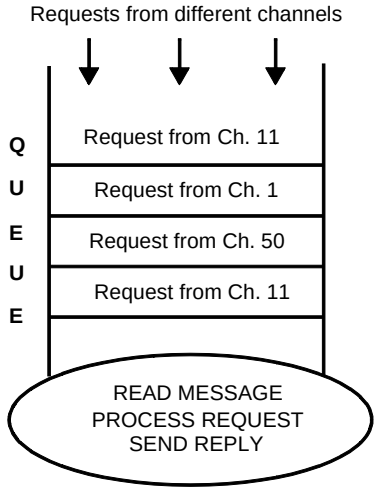
A user-defined function can send error and informational messages to the Transaction Log through calls to `post_log_msg`(message string), which takes a message string as its only argument.

User-defined functions

As shown in Figure 8-3, the application requests to the same User Function process from different channels are not processed until the current request is completed. Multiple requests for the same User Function are placed in a queue and read upon completion of the previous request. A lengthy task can delay processing of other requests in the message queue. For example, if an external device, such as fax or speech recognition, is accessed, a slow task executed by the User Function process can cause other channels to wait long periods of time before their requests are processed.

Consequently, design your user-defined functions to execute quickly in order to prevent delays. The section “Advanced User Function techniques” later in this chapter presents some methods for avoiding these situations.

Figure 8-3: User Function process



Designing a User Function

Before you write any code for your User Function, follow these steps.

1 Decide what you want the User Function to do. Break its activities into separate functions that can be written as individual user-defined function codes. Each USER cell can access only one user-defined code.

2 Choose a name for the User Function. For multithreaded User Functions, the name of the User Function can include a maximum of seven characters. For User Functions that are not multithreaded, the name of the User Function can include a maximum of nine characters. Do not use any of the names listed in the `exe` directory. These names are used by AP software.

3 Decide what buffer information to pass to each user-defined function, and then return to the application. A maximum of 40 input buffers and 40 output buffers can be used each time a USER cell invokes the User Function.

4 Plan your Reply Codes. The User Function returns a Reply Code, which the USER cell uses to go to one of 10 possible branches. Decide which Reply Code the user-defined function returns under various circumstances.

5 Decide how each user-defined function can determine that an error occurred and return an error Reply Code.

6 Plan error messages and informational messages to cover all possibilities. Decide how each user-defined function will make calls to `post_log_msg()` to pass error strings. For more information, see the section “Error handling” earlier in this chapter.

Note: Remove old ".o" object files from the User Function before you recompile using a new release. Specifically, if an object file `usr_main.o` exists from a previous release, the executable that is created from the new release will not function.



CAUTION

Risk of data loss

Nortel recommends that your User Functions do not call `printf()`. Instead, use the error log to avoid interfering with the normal operation of the AP software system administration.

Once you create your design, you can create the User Function.

Coding a User Function

To code a User Function, follow these steps. As you read these instructions, refer to the `usr.c` source code listing in this module.

1 Change directories (cd) to `./usr` directory in the base directory of AP software.

2 Copy and rename the `usr.c` file. Use the same name you chose when you designed the User Function. Refer to the section "Designing a User Function" on page 8-9. For example, to create a User Function called `xfunc`, copy `usr.c` to `xfunc.c`.

3 Edit the switch statement in your `filename.c` file as follows:

- **a** Specify a "case" statement for each user-defined function that the User Function can execute.
- **b** For each case statement, assign a value to the reply_code variable, matching one of the codes in the Reply Code table for your application. This value can also correspond to an error condition; refer to the default case statement in your file.

c To return information to the application from the User Function, assign a value (1 — 40) to the ***p_number_of_output_buffers variable corresponding to the number of output buffers to be passed. Write this statement for each User Function code in your User Function.

d Assign strings to the output buffers in the output_buffer_array structure. Ensure that the values for each buffer, including the NULL terminator, and the total number of characters do not exceed 1920 characters. Data in excess of 1920 characters is truncated. These values are passed back to the application during execution.

4 Save your edits.

Building the User Function

Once you follow the procedure for coding a User Function, you can build the executable file. Follow these steps.

1 At the shell prompt, type the following command, where *usr_func* is the name of your User Function:

```
make -e -f usr.make NAME=usr_func
```

For example, to build an executable file for `xfunc.c`, enter the following:

```
make -e -f usr.make NAME=xfunc
```

Note: Enter the name of your file without the ".c" extension. The executable User Function is `xfunc`, which is created in the current directory.

2 Promote your User Function to the `exe` directory by entering this command:

```
make -e -f usr.make NAME = usr_func promote
```

Note: You must enter this command in order for AP software to run the User Function in the application. Entering this command changes the permission to the executable file and places a binary copy in the directory `exe`.

Testing your User Function with standalone mode

Before you actually use your User Function in an application, you can test the User Function to ensure that it executes properly (for example, each case statement matches a specific user-defined function, or the appropriate number of buffers are passed). You can test the User Function by using the User Function Standalone mode. Follow these steps.

1 Enter the following command at the shell prompt, where *user_func_name* is the name of the User Function you just created:

***usr_func_name* -i**

2 Respond to the following prompts:

Function Code — The number (0 — 99) that identifies the user-defined function and the corresponding switch-case statement in the User Function to test. To exit from the User Function, type -1.

Current Channel — Enter the channel number from which you wish to simulate calling the User Function. If the number is irrelevant, then enter 0.

Timeout value? — Enter the amount of time (0 — 75 seconds) that can pass before the User Function returns and takes the timeout branch. This parameter is used in conjunction with the start_timer and stop_timer functions.

Number of Input Buffers — The number of data buffers (0 — 40) to pass to the User Function.

Input Buffer — Values to be entered into each of the input buffers being used.

The User Function is executed, based on your responses to these prompts, and displays the following information:

User Function Reply — A code returned to the calling application identifying the status of the execution.

Number of Output Buffers — The number of data buffers (0 — 40) that are returned to the calling application.

Output Buffers — The contents of the output buffers.

The following example illustrates the execution of the `usr.c` template in standdalone mode:

```
$ usr -i

User Function ‘usr’ started successfully.

Enter Function Code (-1 to exit): 1
Enter Current Channel: 0
Enter Number Of Input Buffers: 0
Enter timeout duration: 0
==== about to call User Function
Addr = ox7ffff1ac
   Addr = ox7ffff1cc
==== returned from User Function
User Function Reply: 1
Number of Output Buffers: 0

Enter Function Code (-1 to exit): 2
Enter Current Channel: 0
Enter Number Of Input Buffers: 5
Input Buffer #1: 1
Input Buffer #2: 2
Input Buffer #3: 3
Input Buffer #4: 4
Input Buffer #5: 5Enter timeout duration: 0====
about to call User Function
==== returned from User Function
User Function Reply: 2
Number Of Output Buffers: 5
Output Buffer #1: ‘2’
Output Buffer #2: ‘4’
Output Buffer #3: ‘6’
Output Buffer #4: ‘8’
Output Buffer #5: ‘10’

Enter Function Code (-1 to exit): 3
Enter Current Channel: 0
Enter Number Of Input Buffers: 2
Input Buffer #1: 1
Input Buffer #2: 2
==== about to call User Function
==== returned from User Function
User Function Reply: 3
Number Of Output Buffers: 1
Output Buffer #1: ‘3’

Enter Function Code (-1 to exit): 4
Enter Current Channel: 0
```

```
Enter Number Of Input Buffers: 0
==== about to call User Function
==== returned from User Function
User Function Reply: 4
Number Of Output Buffers: 0

Enter Function Code (-1 to exit): 5
Enter Current Channel: 0
Enter Number Of Input Buffers: 0
==== about to call User Function
>>>> call to post_log_msg('Unknown function code:
5')
==== returned from User Function
User Function Reply: 0
Number Of Output Buffers: 0

Enter Function Code (-1 to exit): -1
```

Including the User Function in an application

Once you build a User Function, you can design and build an application to call it. Before you design an application, read the previous chapters and become familiar with the procedures for designing and building applications. Review the description of the USER cell in the *Symposium OPEN IVR Cell Catalog* and follow these steps.

1 Select the name of the User Function using the slider in the User Function Name parameter using the slider.

2 Enter the function code in the Function Code parameter.

3 Enter the number of input buffers in the # of Input Buffers parameter.

4 Enter the number of output buffers in the # of Output Buffers parameter.

5 Enter a timeout value, if applicable.

6 Click each possible Reply Code and the next cell for each in the Reply Code table.

7 Go to the Input Buffer table and list the input buffers.

8 Go to the Output Buffer table and list the output buffers.

Note: The number of input buffers and output buffers are automatically calculated based on the number of input and output buffers you specified.

Once built, you can load and run the application. Refer to Release 4.0 of the *Symposium OPEN IVR System Administration Guide* for procedures, or consult your system administrator. When the application is running, run the Transaction Log Report occasionally to see if the User Function has written any error or informational messages. The *Symposium OPEN IVR System Administration Guide*, documents the procedure for running this report.

usr.c file

This is a listing of the AP software `usr.c` source file, which you can use as a template for creating your User Functions. The `usr.c` file already contains sample User Function code in the switch statement (for example, cases 1, 2, 3, and 4). Remove this code and enter your own case statements.

WARNING

Before you modify the code, copy usr.c to another file and modify the copy.

```
static char usr_c[]="@(#) $Id: usr.c,gen3.02.00 1.12
1996/03/27 22:55:32 Exp $";

/*************************************************
************************
* usr.c
*
* This file contains the routine user_function().
*
*
* WARNING:
*
* Any include files added to this file will NOT be
* reflected in the stated dependencies of usr.makefile.

*************************************************
************************
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#if defined(__cplusplus)
extern “C”

{
#endif
#include “vtk_std.h”
#include “vpf.h”
#include “usr_prototypes.h”

#if defined(__cplusplus)

}
#endif
/*****************************************************
************************
* For multi threaded user functions, the number of child
processes
* should be specified here.
*
* Note that zero children is the default, and means that
the user process
* will operate in a single threaded mode as it has always
done in the past.

*****************************************************
************************
*/
int   number_of_children = 0;  /* number of children to
spawn, if any */

/*****************************************************
************************
*  h-  int   init_user_function(max_chans)
*  int   max_chans;    - maximum possible channels
*
*  DESCRIPTION
```

```
*  User defined user_function initialization
*
* This initialization routine is for the user to perform
certain
* tasks before the process begins taking requests from
applications.
*
* Note that in a multi-threaded user function. This
routine is executed
* in each child process, but not in the parent (manager)
process
*
* RETURNS:  TRUE  if successful
*             FALSE  otherwise
**************************************************
************************
*/
int init_user_function(int max_chans)

{
verbose_msg (“init_user_function”);
return(TRUE);

}

/**************************************************
************************
* -h-  void  end_user_function(max_chans)
*  int   max_chans;    - maximum possible channels
*
* DESCRIPTION
* User defined user_function cleanup
*
* This termination routine is for the user to perform
* certain tasks after the process has been notified to
* terminate.
*
*  Note that in a multi-threaded user function. This
routine is executed
```

```
* in each child process, but not in the parent (manager)
process

*
**************************************************
************************

*/
void end_user_function(int max_chans)

{

 verbose_msg (“end_user_function”);

 return;

}

/*-------+---------+---------+---------+---------+----
-----+---------+--------*/

/*-h-  int   user_function (function_code,
current_channel,

**  number_of_input_buffers, input_buffer_array,

** p_number_of_output_buffers, output_buffer_array)
** int   function_code;
** int   current_channel;
** int   number_of_input_buffers;
** char
input_buffer_array[BUFS_PASSED][MAX_VAR_BUF_SIZE+1];
**   int   *p_number_of_output_buffers;
**   char
output_buffer_array[BUFS_PASSED][MAX_VAR_BUF_SIZE+1];
```

```
**   u_long tout;
**
** DESCRIPTION:
**   Perform the actions of each user function.
**   Each CASE in the SWITCH statement acts as a
different user function
**   within this process
**
** RETURNS:
**   The reply code (0 - 9) to the user function cell.
**   The cell will then branch based on this value.

**************************************************
************************
*/
int user_function(int function_code,

    int current_channel,
    int number_of_input_buffers,

    char
input_buffer_array[BUFS_PASSED][MAX_VAR_BUF_SIZE+1],
    int *p_number_of_output_buffers,

    char
output_buffer_array[BUFS_PASSED][MAX_VAR_BUF_SIZE+1],

    u_long tout)

{

 /* -------- Variables for use by ALL user functions.
------------------ */

 int    reply_code = 0;
 char  error_message[80];

/* -------- Variables used by THIS user function.
--------------------- */
 int   count;
 long  sum;

/*
```

```
* Assign a default value for the number of output
buffers.
* (Because p_number_of_output_buffers is a pointer to an
integer the
* indirection operator (‘*’) must be used to assign an
integer value.)
*/
*p_number_of_output_buffers = number_of_input_buffers;

switch(function_code)

 {

 case 1:
  /* Do nothing */
  *p_number_of_output_buffers = 0;
   reply_code = 1;
   break;

case 2:

  /* Numerically double buffer values */
  for (count=0; count<number_of_input_buffers; count++)

  {
   sprintf(output_buffer_array[count], “%ld”,

      2*atol(input_buffer_array[count]));

  }
  *p_number_of_output_buffers =
number_of_input_buffers;
   reply_code = 2;
   break;

case 3:

  /* Numerically sum buffer values */

  sum = 0L;
  for (count=0; count<number_of_input_buffers; count++)

  {
   sum = sum + atol(input_buffer_array[count]);
```

```
  }
  sprintf(output_buffer_array[0], “%ld”, sum);
  *p_number_of_output_buffers = 1;
  reply_code = 3;
   break;

case 4:

  /* Any instruction(except those listed below) will be
interrupted when a timeout occurs. The timer_expired
variable will be set to TRUE. The User function
programmer needs to check timer_expired after every
instruction that can block the process as the example
below shows. */

  /* Note: The UNIX system() call or any child of this
user
function will not be able to receive the timeout signal
*/


  start_timer(tout);   /* tout is timeout duration */
  sleep(180);        /* sleep for 3 minutes */

 /* A suspected timeout may occur. Check global variable
timer_expired and call timeout_function if timeout
occurs */

  if (timer_expired)
{

   timeout_function(function_code, current_channel,
   number_of_input_buffers, input_buffer_array,
   p_number_of_output_buffers, output_buffer_array);
   return usr_timeout_code();

  }
*p_number_of_output_buffers = 0;
stop_timer();
reply_code = 4;
break;
```

```
default:
/* -------- Print Error if in verbose mode
-------------------- */
if (fp_verbose)

   fprintf(fp_verbose, “\tfunction_code %d not
implemented\n”,

       function_code);

 /* -------- Report Error to error logger
---------------------- */

  sprintf(error_message, “Unknown function code: %d”,
function_code);

  post_log_msg(error_message);

  *p_number_of_output_buffers = 0;

  break;
}
return (reply_code);
}

/*************************************************
************************
*-h-  int   init_user_manager(max_chans)
*    int   max_chans;    - maximum possible channels
*
* DESCRIPTION
*   User defined user_manager initialization
```

```
*
* This initialization routine is for multi-threaded user
functions
* where the user needs to perform certain tasks within
the manager
* process before it begins taking requests from
applications
* or communicating with its children user functions.
*
* Note that this routine only executes if
number_of_children > 0
*
* RETURNS:  TRUE  if successful
*         FALSE  otherwise


*****************************************************
************************
*/
int init_user_manager(int max_chans)
{
verbose_msg (“init_user_manager”);

return(TRUE);

}
```

```
/*****************************************************
************************
*-h-  void  end_user_manager(max_chans)
*    int   max_chans;    - maximum possible channels
*
* DESCRIPTION
* User defined user_manager cleanup
*
* This termination routine is for multi-threaded user
functions
* where the user needs to perform certain tasks within
the
* manager process after it has been notified to
terminate.
* Note that this routine only executes if
number_of_children > 0 and
*   is only executed in the parent (manager) process.

*
*****************************************************
************************
*/
void end_user_manager(int max_chans)

{
verbose_msg (“end_user_manager”);
return;

}

/*****************************************************
************************
* -h- void timeout_function(function_code,
current_channel,
* number_of_input_buffers, input_buffer_array,
* p_number_of_output_buffers, output_buffer_array )
* DESCRIPTION
*  Function to execute when timeout occurs.
*
* RETURN: nothing
```

```
*****************************************************
**********************
*/
void timeout_function(int function_code, int
current_channel, int number_of_input_buffers,
char input_buffer_array[BUFS_PASSED][MAX_BUF_SIZE+1],
int *p_number_of_output_buffers,
char output_buffer_array[BUFS_PASSED][MAX_BUF_SIZE+1])

{
printf("Timeout function executed!\n");
}
```

Advanced User Function techniques

If you are familiar with writing User Functions, refer to the useful techniques described in the remainder of this chapter.

Task Manager

If you find the performance of your User Function unacceptable, you can distribute its processing load. The Task Manager feature distributes incoming User Function requests to one of several concurrent processes (children) instead of acting on each request sequentially.

This significantly reduces the potential bottleneck associated with a User Function that interacts with a slow resource, such as a device or large database. Note that child processes cannot exchange data with each other, only with AP software.

The maximum number of children that you can use to defer load is 10. While the system does not limit the number of applications using a User Function, only 10 channels can use a specific User Function simultaneously. Each of the 10 child processes handles a single channel request in this situation. All others will wait in a queue.

> ***Note:*** Degradation can occur when the User Function uses more than five concurrent processes.

To designate the number of `user_function` children, edit the variable `number_of_children` at the top of the `usr.c` file. The default is `0`, which creates a single-threaded environment: that is, one that spawns no children and processes all requests itself.

You can also include two optional functions in your C code:

- `init_user_manager()`, which performs initialization once at the creation of the User Function Task Manager, when the first application that uses the User Function is loaded.
- `end_user_manager()`, which performs shutdown tasks once in the User Function Task Manager, when the last application that uses the User Function is unloaded.

You can use the following two functions in the single-threaded environment, but when used in conjunction with a Task Manager with concurrent processes, the functions take on the following specialized purposes:

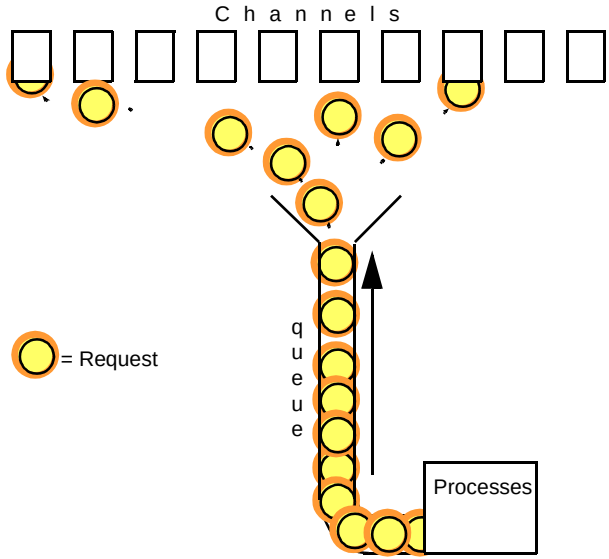
- `init_user_function()` performs initialization once at the creation of each User Function child process when the first application is loaded.
- `end_user_function()` performs shutdown tasks in the User Function child process once when the last application is unloaded.

The dispatching algorithm used by the Task Manager guarantees that, while a child concurrent process can serve more than one channel, it will consistently serve the same set of channels.

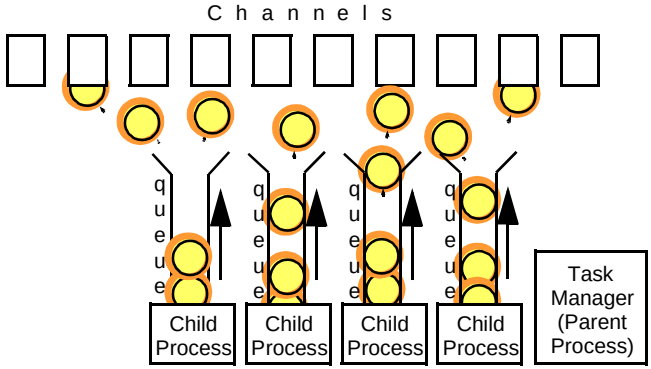
Since a child may need to maintain information about calls associated with each channel, the child only needs to dynamically allocate memory for those channels with which it is associated. Therefore, when using multiple child processes (five, for example), each child is responsible only for its part of the memory allocation (in this case, 20 percent). See Figure 8-4, which illustrates the Task Manager.

Figure 8-4: Task Manager

Single-Threaded



Task Manager



Timeout

You can set a parameter in your User Function to follow a timeout branch if its execution time exceeds the timeout value specified on the parameter page of the USER cell.

To implement this functionality, enter a `start_timer(duration)` function in your `user_function` at the beginning of the code you wish to bind with a time limit. If the code you limit exceeds the duration specified, then the system executes a handler function, which sets the global variable `timer_expired` to `TRUE`. The instruction following the code that could exceed the time limit should contain a test for this variable. Include `stop_timer` in the `user_function` to shut off the previously requested timer.

If the `timer_expired` variable is `TRUE`, you can call the function `timeout_function`, which contains code you have specified to clean up from a timeout condition before a reply is returned to the application.

Example

For example, to open a connection to an external device and cancel it if it exceeds a specified time limit, include the following lines in your `user_function`:

```
start_timer(tout); /* tout is timeout duration */
open_device(....); /* this represents any operation you
wish to limit */
if (timer_expired)
/* Check global variable. TRUE means a timeout occurred.
*/
{
    /* to clean up */
timeout_function(function_code,
current_channel,
number_of_input_buffers,
input_buffer_array,
p_number_of_output_buffers,
output_buffer_array);
   return  usr_timeout_code();
}
stop_timer();
/* cancel the timer if connection made within the time
allowed*/
```

> ***Note:*** The timeout signal does not interrupt UNIX system() calls or any child processes of the User Function.

Use the `timeout_function()` to set up return values to the application. The timeout reply message is sent elsewhere.

Customizable user cells

AP software enables you to create your own cells, based on the OPEN IVR USER cell, and modify them by means of the Customizable User Cell Builder (CUCB). You can then use these cells in the Application Editor.

> ***Note:*** Use the Cell Builder to create a customized icon that you can select from your cell palette and place on the drawing board. Code the actual functionality for the cell in a separate `usr.c` file.

This feature is useful for customizing AP software to facilitate specialized application development. You can customize USER cells to interface with proprietary processes developed by your own User Functions and perform complex calculations on your customized algorithms.

You can customize USER cells to build Computer Telephony Integration (CTI) functions into your applications in order to interface with switches at the Central Office (CO), thereby allowing your application to activate and deactivate features at the CO level.

You can also customize USER cells to interface your voice mail application with your SQL database in order to access billing provisioning features to update the class of service provided to a user.

Customizable USER cell builder

To customize a USER cell, invoke the Cell Builder by entering the following command on the command line:

```
cucb [-fbw] [-help] [-load file] [-r]
```

where

- `fbw` forces the graphics to appear in black and white.
- `help` generates usage messages.
- `load <file>` loads a cell description file for editing.
- `r` displays the current revision level.

When you invoke Cell Builder without specifying a file, it displays a blank form that contains the same default values as the USER cell. Create your own cell by modifying these values.

Some fields in this form must contain token strings. A token is a string that corresponds with an entry in the `xae.lng` file so that you can internationalize your applications, if desired.

Note: If the token does not already exist in this file, enter both the token and its value in the xae.lng file before your application that uses that cell is saved, or the Application Editor cannot load the application. The xae.lng file resides in the platform language area (usually ../ui_lang/en_US). For further information on entering the value for a token, refer to Appendix B, "Customization and internationalization."

After you set all the fields to the desired values, save the cell in the IVR Developer cell area (usually ../cells). Once you save the cell, restart the IVR Developer in order to use the cell in an application.

Note: If you change the order of input values, reflect these changes in the cell's server as well.

When you invoke Cell Builder, a menu bar appears at the top of the Cell Builder main window displaying both a File pulldown and a Help pulldown. The File pulldown presents the following options:

New

To start editing a new customized user cell not yet associated with a cell description file (`.cdo` file). When you select this command, the system automatically closes any currently open cell description file (`.cdo` file). If you made unsaved changes to another existing `.cdo` file when you selected the New command, the system will prompt you to save your current cell.

Open

To open an existing cell description file (`.cdo` file). A file browser appears and prompts you to select the file you want. When you select this command, the system automatically closes any currently open cell description file (`.cdo` file). If you made unsaved changes to another existing .cdo file when you selected the Open command, the system will prompt you to save the other file.

Save

To save your current work. If you are editing a cell description file (`.cdo` file), then the current cell will be saved in that file. Otherwise, a file browser appears and prompts you to designate the name of the file in which you want to save your work. That file becomes the active file for further saves.

Save as

To specify a new cell description file (`.cdo` file) into which to save the current cell. A file browser appears and prompts you to designate the name of the file in which you want your work saved. That file becomes the active file for further saves.

Close

To start editing a new customized user cell not yet associated with a cell description file (`.cdo` file). When you select this command, the system automatically closes any currently open cell description file (`.cdo` file). If you have made unsaved changes to another existing `.cdo` file when you select the Close command, the system will prompt you to save your current cell.

> ***Note:*** For this version of Cell Builder, the Close and New commands are identical.

Exit

To terminate the current editing session. If you have unsaved changes in your file when you select the Exit command, the system will prompt you to save before exiting.

The Help pulldown presents the following options:

Overview

To access a dialog box containing general help information on the purpose and use of the Cell Builder.

About

To access a dialog box containing product and copyright information.

The Cell Builder cell editing form contains the following field, which you can customize in order to define your cell:

Table 8-1: Cell builder cell editing form fields

Field	Description
Cell Type Name	Enter the token name you want to use for the cell's short and long names.
Cell Icon	Enter the file name that contains the icon pixmap from the `/ui_lang/en_US` directory for your customized cell. If you designate the same Cell Icon for multiple cells that you create, the cells will appear in the same cell group on the Cell Palette under that particular icon.
Display Timeout?	Change the timeout value for this cell. Select the toggle button in this field.
Timeout	Indicate the timeout period. If Display Timeout is set to true, set this slider to the desired default timeout value; if Display Timeout is set to false, set this slider to the fixed timeout value for this cell.
Function Name	Enter the text string name of the User Function you want to associate with this cell.
Display Function Code?	Allows you to change the function code for this cell. Select the toggle button in this field appropriately.
Function Code	Indicates the (default) function code for this cell. Adjust the slider in this field appropriately.

Function Code Range	Indicate the range of function codes. Set the range by adjusting the two sliders in this field. If the Display Function Code toggle is set to off, you cannot adjust these sliders. ***Note:*** The two fields Function Code and Function Code Range are interdependent. Therefore, a change in one may cause a change in the other.
Input Buffers or Values Title	Specify the token name for the title that is displayed above the Input Buffers or Values field in the parameter page for the cell.
Number of Input Buffers or Values	Designate the number of input buffers or values for this cell. Set the slider in this field. To edit the labels for input buffers or values, click the Edit Labels push button in this field. A dialog box appears in which you can enter the token names for each of the input buffers or values. (See the section “Buffer Labels dialog box, that follows.)
Output Buffers Title	Specify the token name for the title that appears above the output buffers field in the Parameters page of the cell.
Number of Output Buffers	Designate the number of output buffers for this cell. Set the slider in this field. To edit the labels for output buffers, click the Edit Labels push button in this field. A dialog box is displayed in which you can enter the token names for each Output Buffer. (See the section on “Buffer Labels dialog box, that follows.)
Timeout Branch	Enter the token name for the Timeout Branch label in the cell.
Fail Branch	Enter the token name for the Fail Branch label of the cell.
Reply Codes	Select all the reply codes for the cell. Unlike the reply codes in the USER cell, you cannot change the reply codes in the customizable user cell once you set them.

Buffer Labels dialog box

In the Buffer Labels dialog box, you can enter the token names for the labels you want to display to the left of each buffer in the cell Parameters page. The default token values remain in any unchanged fields.

> ***Note:*** Although the buffer numbers start at zero, the default label numbers start at one.

To revert to the label values, follow these steps.

1. If you want to revert to the label values in effect when you requested the Buffer Labels dialog box, click the Cancel button.
2. Click OK when you are satisfied with the labels displayed in the dialog box.

> ***Note:*** To use tokens in your customized cell that do not currently exist in the `xae.lng` file, names for the tokens in that file. The `xae.lng` file is located in the o-s language area: (usually in the directory `../ui_lang/en_US`).

Customizing a user cell

You can create a cell for a frequently accessed User Function by customizing a USER cell with information specific to your User Function, naming this new cell and placing the cell on your cell palette. With Cell Builder, you can specify a customized icon, which you can select from your cell palette, and place the icon on the drawing board. Code the functionality for the cell in a separate `usr.c` file.

To customize a USER cell in order to access a User Function, follow these steps.

> ***Note:*** Before customizing your USER cell, read the sections in this chapter about creating User Functions.

1. Create and save your cell definition file.
2. Specify the tokens in Cell Builder to label and identify the cell.
3. Identify the group on the Cell Palette where you want the cell to appear. If an appropriate group does not currently exist, create one.
4. Create a Help File for the new cell.

Note: If you edit your User cell after you save an occurrence of the cell in an application, this previously placed cell will not reflect the new functionality. However, if you subsequently open the application and place another copy of the cell in a different spot in the call flow, the new instance of the cell includes changes.

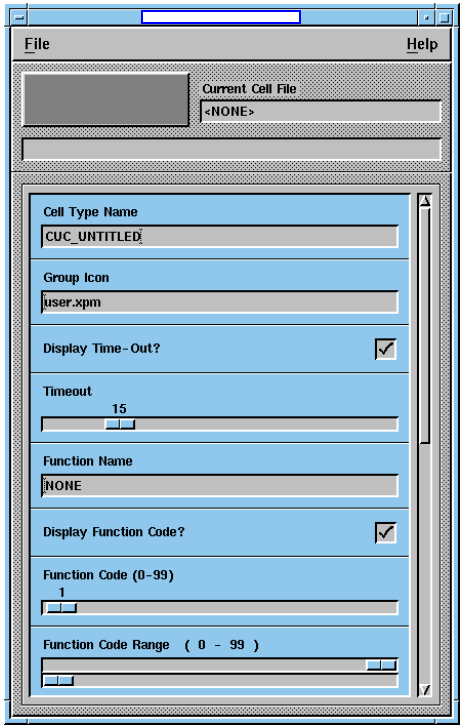
In order to update the functionality of the cell placed in the previous editing session, delete the cell at its location and place it again in the palette.

Create and save your cell

When you invoke Cell Builder without specifying a file, a form is displayed that contains the same default values as the USER cell. Create your cell by modifying these values with the Cell Builder in order to create a new cell description file. Figures 8-5 and 8-6 show a user-modified form (top and bottom halves) provided by Cell Builder.

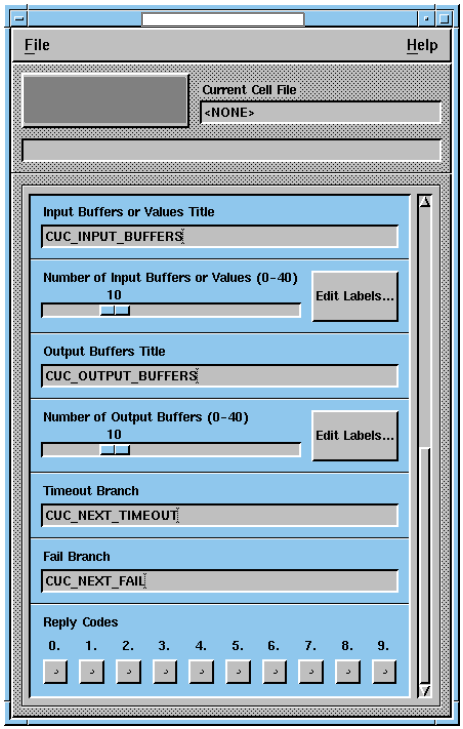
Note: The `Cell Type Name` and `Group Icon` fields in Figure 8-5 are edited for this customized user cell, and the `Current Cell File` field contains the name of the file in which you saved the cell.

Figure 8-5: Top half of the blank form provided by Cell Builder



Note: The `Reply Codes` field in Figure 8-6 is edited for this customized cell.

Figure 8-6: Bottom half of the blank form provided by Cell Builder



When you save this file, Cell Builder appends its name with `.cdo`. The IVR Developer searches the `$VOICE_HOME/cells` directory for the files with the `.cdo` extension in order to determine the contents of your cell's Parameters page whenever an application contains any of these cells. Always save your `.cdo` file in this directory.

Specifying tokens in Cell Builder

When you call up a new cell description file (`.cdo file`), specify the tokens for the cell type and function name. If you wish to change the defaults labels that appear on the cell Parameters page and on the cell icons once they are placed on the drawing board, edit these fields on the Cell Builder form. Then define any tokens that you edit in the file `xae.lng`. See Figure 8-7.

Define both the token and its value in the xae.lng file before you save the application that uses the cell. Otherwise, the Application Editor cannot load an application using your cell. Prefix any tokens that you use in the Customizable User Cell definition with CUC. Otherwise, Cell Builder adds the prefix. When you name your tokens, ensure that you or Cell Builder do not inadvertently redefine any pre-existing tokens in the xae.lng file.

The xae.lng file is located in the AP software language directory, usually $VOICE_HOME/ui_lang/en_US. See Appendix B, "Customization and internationalization," for further information on entering the value for a token.

> ***Note:*** When editing the xae.lng file, be sure to press <Return> at the end of each line, including the last line.

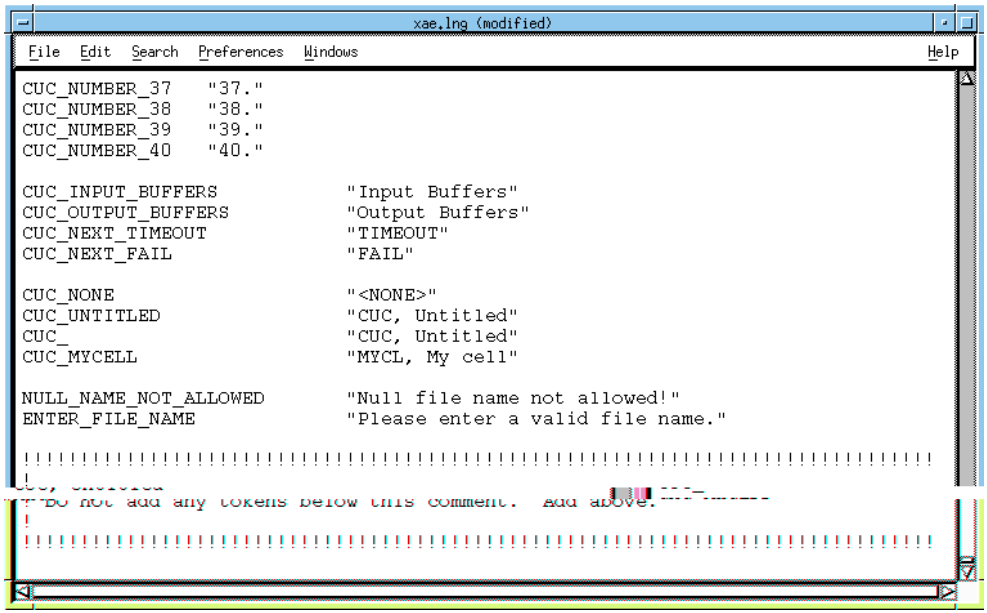
Define the cell type token as a comma-separated pair of strings, where the first string is the short name for the cell type (that contains a maximum of seven characters), usually in upper case characters. The second string is the long name for the cell type. The size is not limited, but visually truncated to about 15 characters on the face of the cell, depending on the font used, with usually only the first character uppercase.

Figure 8-7 presents an xae.lng file that includes the following line:

```
CUC_MYCELL  "MYCL, My cell"
```

This line defines the token, CUC_MYCELL, and its value, "MYCL, My cell".

Figure 8-7: xae.lng file for a customized cell



Identify the group for the cell on the Cell Palette

In order for your cell to appear on your Application Editor Cell Palette in an existing cell group, specify the icon file name of the desired group in the appropriate field of your cell's Cell Builder form. These group icon files are located in the language directory, usually `$VOICE_HOME/ui_lang/en_US`. The default group icon file includes the USER cell, `user.xpm`.

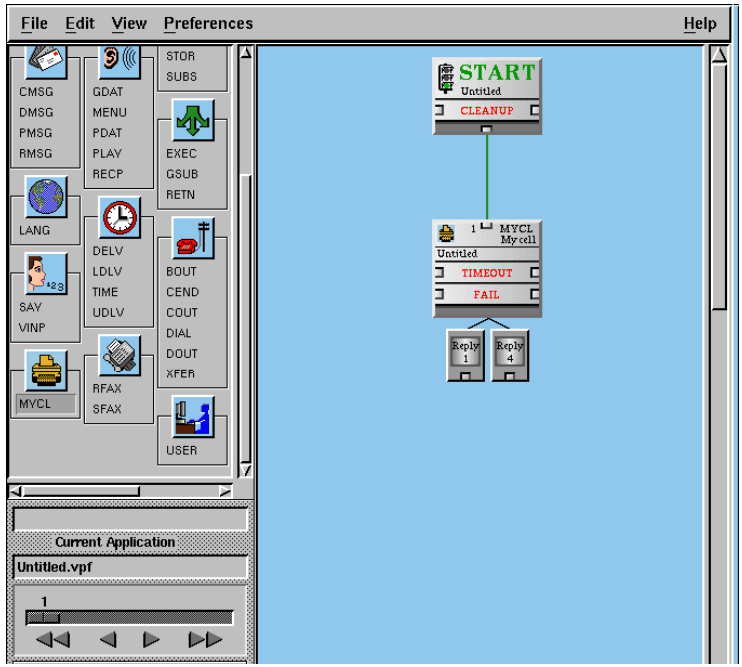
Alternatively, you can create a new group for your cell. You also create group icons for the new group.

Include in each group a large and a small icon. The large icon, used to represent your group on the Cell Palette, must be 32x32 pixels. Include this icon in the file that you specified in the `Group Icon` field on the Cell Builder form.

The small icon, used on the drawing board in any cell created from your group, must be a 16x16 pixel icon. Follow the current convention and create the small icon as simply a half-scale version of the large icon.

Figure 8-8 shows the User Defined cell MYCL under its own large icon on the Cell Palette, and on the drawing board with the small icon in the upper left corner of the cell.

Figure 8-8: User defined cell MYCL on cell palette and drawing board



If the large icon file is called, for example, `my_group.xpm`, then name the small icon `my_group16.xpm` to ensure that the two icon files adhere to the appropriate naming conventions.

To create your icons, access a bitmap editor that can edit or export to XPM-format files.

If you choose to use transparency, then use a bitmap editor that outputs human-readable XPM3 format files. Note that you are not required to use transparency.

Note: A public domain bitmap editor, called pixmap, is located on a shareware site, that supports transparency and outputs in XPM3 format.

When you create your icons, use the same color for all your background pixels. The pixmap editor allows you to set the symbolic name for the background color to `None`. If you do not use the pixmap editor and edit your XPM3 format file with a text editor, then check the text representation of your icon and determine which character (possibly a space, period, single quote, or letter of the alphabet) represents the background color. Find the line in the preamble that defines color for that character. Change the color for the background character to `None`.

For example, `user.xpm` begins as follows:

```
/*XPM*/
static char *noname [] = {
/* width height ncolors chars_per_pixel  */
“16 16 7 1”,
/* colors */
“‘  c black m black”
“a c dark blue m white”,
“b c gray33 m white”,
“c c gray53 m white”,
“d c brown m white”,
“e c blue m black”,
“f c None m None”,
/* pixels */
“ffffffffffffffff”
“f’’’’fffffffffff”
“‘c’fffffffeeef”
“bc’f’fffffeeeea”
```

Note that the character `f` is used for the background pixels. The following line defines the color and the monochrome values for the pixels represented by the character `f` as `None`:

```
“f c None m None”,
```

Once you create the XPM files, move the files to the AP software language directory, which is usually `$VOICE_HOME/ui_lang/en_US`.

Creating a help file for the new cell type

Create a help file for each new cell type.

To create the file, follow these steps.

1. Enter the information for your new cell type in an ASCII file using a maximum of 80 characters per line; do not use tab characters.
2. Save this file with the same token name as the cell type, and add the `.hlp` extension.

   For example, ensure that you define a cell with the cell type token defined as follows:

   ```
   CUC_MYCELL“MYCL, My cell”
   ```

3. Next, save the help text for this type of cell in the file in the AP software language directory:

   ```
   CUC_MYCELL.hlp
   ```

4. Store your new Help file in the AP software language directory, usually `$VOICE_HOME/ui_lang/en_US`.

To access this help file, the user can click on the `Help` push button on the Parameters page of the cell.

Appendix A: Standard prompts

The following is a list of standard OPEN IVR prompts. Standard prompts are used in many applications. Recordings of these prompts are included with OPEN IVR.

Prompt number	Prompt
0	zero
1	one (masculine version)
2	two
3	three
4	four
5	five
6	six
7	seven
8	eight
9	nine
10	ten
11	eleven
12	twelve
13	thirteen
14	fourteen

Prompt number	Prompt
15	fifteen
16	sixteen
17	seventeen
18	eighteen
19	nineteen
20	twenty
21	thirty
22	forty
23	fifty
24	sixty
25	seventy
26	eighty
27	ninety
28	hundred
29	thousand
30	oh
31	star
32	pound
33	January
34	February
35	March
36	April
37	May

Prompt number	Prompt
38	June
39	July
40	August
41	September
42	October
43	November
44	December
45	on
46	at (singular)
47	a.m.
48	p.m.
49	The system storage is nearing capacity.
50	The system storage is at capacity.
51	We were unable to record this message.
52	You cannot record any new messages at this time.
53	Welcome to...
54	Hello.
55	Thank you. Goodbye
56	That is an invalid response.
57	There are...
58	The number of messages waiting is...
59	At the tone,leave your message. When finished, press...

Prompt number	Prompt
60	Your message has been delivered.
61	We were unable to deliver your message.
62	No new messages may be recorded at this time.
63	There are no more messages waiting.
64	The operator will be with you momentarily.
65	You have paused the recording. To resume, press...
66	To save, press 1. To review, press 2. To delete, press 3.
67	To rerecord, press 1. To continue, press 2.
68	To review, press 1.
69	Please enter the system password.
70	This message was sent on...
71	Please enter your password.
72	Please hold.
73	That is an invalid password.
74	We are temporarily unable to access this message.
75	You have no messages waiting.
76	The newly recorded prompt is...
77	The message has been deleted.
78	million
79	billion
80	dollar

Prompt number	Prompt
81	dollars
82	cent
83	cents
84	area code
85	and
86	by
87	.2 seconds of silence
88	.4 seconds of silence
89	.8 seconds of silence
90	up
91	down
92	half
93	third
94	fourth
95	fifth
96	sixth
97	seventh
98	eighth
99	ninth
100	tenth
101	point
102	points
103	thirds

Prompt number	Prompt
104	fourths
105	fifths
106	sixths
107	sevenths
108	eighths
109	ninths
110	tenths
111	sixteenths
112	thirty-seconds
113	sixty-fourths
114	weekdays
115	Resume recording at the tone.
116	At the tone, leave your message.
117	Recording has been aborted. The message will not be saved.
118	When finished, press...
119	(Not used.)
120	(Not used.)
121	A
122	B
123	C
124	D
125	E

Prompt number	Prompt
126	F
127	G
128	H
129	I
130	J
131	K
132	L
133	M
134	N
135	O
136	P
137	Q
138	R
139	S
140	T
141	U
142	V
143	W
144	X
145	Y
146	Z
147	Sunday
148	Monday

Prompt number	Prompt
149	Tuesday
150	Wednesday
151	Thursday
152	Friday
153	Saturday
154	twenty-one
155	twenty-two
156	twenty-three
157	twenty-four
158	twenty-five
159	twenty-six
160	twenty-seven
161	twenty-eight
162	twenty-nine
163	thirty-one
164	thirty-two
165	thirty-three
166	thirty-four
167	thirty-five
168	thirty-six
169	thirty-seven
170	thirty-eight
171	thirty-nine

Prompt number	Prompt
172	forty-one
173	forty-two
174	forty-three
175	forty-four
176	forty-five
177	forty-six
178	forty-seven
179	forty-eight
180	forty-nine
181	fifty-one
182	fifty-two
183	fifty-three
184	fifty-four
185	fifty-five
186	fifty-six
187	fifty-seven
188	fifty-eight
189	fifty-nine
190	sixty-one
191	sixty-two
192	sixty-three
193	sixty-four
194	sixty-five

Prompt number	Prompt
195	sixty-six
196	sixty-seven
197	sixty-eight
198	sixty-nine
199	seventy-one
200	seventy-two
201	seventy-three
202	seventy-four
203	seventy-five
204	seventy-six
205	seventy-seven
206	seventy-eight
207	seventy-nine
208	eighty-one
209	eighty-two
210	eighty-three
211	eighty-four
212	eighty-five
213	eighty-six
214	eighty-seven
215	eighty-eight
216	eighty-nine
217	ninety-one

Prompt number	Prompt
218	ninety-two
219	ninety-three
220	ninety-four
221	ninety-five
222	ninety-six
223	ninety-seven
224	ninety-eight
225	ninety-nine
226	hundreds
227	five hundred
228	thousands
229	ten thousands
230	hundred thousands
231	ones
232	year
233	days
234	of
235	millions
236	billions
237	minus
238	negative
239	first
240	second

Prompt number	Prompt
241	eleventh
242	twelfth
243	thirteenth
244	fourteenth
245	fifteenth
246	sixteenth
247	seventeenth
248	eighteenth
249	nineteenth
250	twentieth
251	thirtieth
252	percent
253	left square bracket
254	backslash
255	return
256	enter
257	tab
258	backspace
259	delete
260	space
261	exclamation mark
262	period
263	ampersand

Prompt number	Prompt
264	apostrophe
265	plus
266	comma
267	dash
268	divide
269	colon
270	semicolon
271	less than
272	equal
273	greater than
274	question mark
275	open parenthesis
276	close parenthesis
277	asterisk
278	caret
279	underbar
280	single quote
281	double quote
282	right square bracket
283	tilde
284	pipe
285	open brace
286	close brace

Prompt number	Prompt
287	dollar sign
288	at symbol
289	celsius
290	fahrenheit
291	degrees
292	above
293	below
294	centigrade
295	degree
296	hour
297	minute
298	one (feminine version — French, German, Italian, Portuguese, Spanish)
299	twenty-one (feminine version — French, Italian, Portuguese)
300	eins (German)
301	one hundred (Portuguese, Spanish)
302	two hundred (Portuguese)
303	three hundred (Portuguese)
304	four hundred (Portuguese)
305	twenty-two (feminine version — Portuguese)
306	six hundred (Portuguese)
307	seven hundred (Portuguese)

Prompt number	Prompt
308	eight hundred (Portuguese)
309	nine hundred (Portuguese)
310	two (feminine) — (Portuguese)
311	hours (Portuguese) o'clock (Japanese)
312	minutes (Portuguese)
313	months (Mandarin) month (Japanese)
314	over (Mandarin, Cantonese)
315	at (plural — Portuguese)
316	hundred million (Mandarin, Cantonese)
317	day (Japanese)
318	point (fraction — Mandarin, Cantonese, Japanese)

Appendix B: Customization and internationalization

You can customize the appearance of the OPEN IVR user interface to

- Alter the appearance of interface elements.
- Translate OPEN IVR into foreign languages.
- Alter the wording of menu options.

The changes you make will take effect immediately. No recompilation of source code is necessary.

Modifying resource files and colors

The `Snap` and `Snap_bw` resource files contain the colors used by OPEN IVR processes. Window manager and process-independent color resources are stored in the `.Xdefaults` file.

As supplied, these files support 16-color and black-and-white systems (see Table B-1). On displays that support more colors, you may wish to alter the appearance of the user interface by modifying the color entries in these files.

Should you decide to modify the supplied resource files, select colors from the palette that follows or your changes may fail to work on systems that support only 16 colors.

Table B-1: The Nortel 16-color palette

white	red
light blue	peach puff
medium blue	magenta4
blue	light gray

forest green	gray33
yellow	gray53
orange	gray75
brown	black

User interface language directory

The user interface language directory, referred to here as `ui_lang`, supports the customization/internationalization feature. You can customize items (such as icons, user interface language files, error string files, and help text) in the `ui_lang` directory for a particular language or special release.

An environment variable, `VOICE_LANG`, assigned within the user's `.profile`, points to the active `ui_lang` directory. Nortel supplies the language files in U.S. English.

> ***Note:*** If you want to set the language to something other than U.S. English, you must modify the profile in Nortel user's directory and the `gen` script file in `/etc`.

If the OPEN IVR installation directory as called `/u/nortel,` then the following `.profile` line would tell OPEN IVR to look inside `/u/nortel/ui_lang/en_US` for language-specific files:

```
VOICE_LANG=en_US;export VOICE_LANG
```

If multiple languages are required for a project, you can create several `VOICE_LANG` subdirectories to support them. Suggested choices for naming subdirectories appear in Table B-2.

Table B-2: Languages and subdirectories

Language	Country	Subdirectory
Danish	Denmark	da_DK
Dutch	Belgium	nl_BE
Dutch	Netherlands	nl_NL
English	Japan	en_JP
English	U.K.	en_GB
English	U.S.	en_US
Finnish	Finland	fi_FI
French	Belgium	fr_BE
French	Canada	fr_CA
French	France	fr_FR
French	Switzerland	fr_CH
German	Germany	de_DE
German	Switzerland	de_CH
Icelandic	Iceland	is_IS
Italian	Italy	it_IT
Japanese	Japan	ja_JP
Korean	Korea	ko_KR
Norwegian	Norway	no_NO
Portuguese	Portugal	pt_PT
Spanish	Spain	es_ES
Swedish	Sweden	sv_SE



CAUTION

Risk of data loss

You can modify tokens without risking overwriting the file at installation by creating a new `xae.lng` file under the `$VOICE_HOME/ui_lang/"$VOICE_LANG"_site` directory.

When the editor is launched, it will compare the `xae.lng` files in both `$VOICE_HOME/ui_lang/$VOICE_LANG` and `$VOICE_HOME/ui_lang/"$VOICE_LANG"_site` and choose or include information according to the following guidelines:

1 The editor will select the value of a token defined under the `$VOICE_HOME/ui_lang/"$VOICE_LANG"_site` directory over a standard value given in the `$VOICE_HOME/ui_lang/$VOICE_LANG` directory. These files are read in alphabetical order so that when two files reference the same token, the last alphabetically listed file will take precedence.

2 If the token is defined under the `$VOICE_HOME/ui_lang/"$VOICE_LANG"_site` directory has no corresponding entry in the `$VOICE_HOME/ui_lang/$VOICE_LANG` directory, then the editor will include that token and definition along with those that exist in the original `xae.lng` file.

An example of this process follows. In this example, OPEN IVR will overwrite any tokens found in the USER FILES that match with the SYSTEM FILE (`xae.lng`). OPEN IVR will then append all remaining tokens to the token table.

The only system file affected by this change is the `xae.lng` file. You still must modify files in the system language directory. Note that you can still modify the language files located in the system directory (`$VOICE_HOME/ui_lang/$VOICE_LANG`). You must also maintain the user directory with your token modifications.

SYSTEM FILE:
$VOICE_HOME/ui_lang/$VOICE_LANG/xae.lng

USER FILE:
$VOICE_HOME/ui_lang/"$VOICE_LANG"_site/xae.lng
$VOICE_HOME/ui_lang/"$VOICE_LANG"_site/xae_CELLS.lng
$VOICE_HOME/ui_lang/"$VOICE_LANG"_site/xae_COLORS.lng
$VOICE_HOME/ui_lang/"$VOICE_LANG"_site/xae_ERRORS.lng
$VOICE_HOME/ui_lang/"$VOICE_LANG"_site/xae_WARNINGS.lng

For example, if the following SYSTEM FILE `xae.lng` includes the token

TOKEN_A “HELLO”

and the USER FILE in the site directory `xae.lng` includes the token

TOKEN_A “GOODBYE”

then the site value will take precedence over any reference of TOKEN_A in the SYSTEM FILE, and TOKEN_A will take the value “GOODBYE.”

Items in the UI_LANG directory

Application Editor button label icons

The files in Table B-3, located in the `UI_LANG` directory, contain the icons used in the Application Editor Cell Palette as button labels. All icons are encoded in XPM (X Pixmap) format. These icons use colors from the Nortel 16-color palette.

Table B-3: Application Editor icons

adsi.xpm	intnatnl.xpm
adsi16.xpm	intnatnl16.xpm
audio.xpm	logo.xpm
audio16.xpm	message.xpm
buf_manip.xpm	message16.xpm
buf_manip16.xpm	msg_db.xpm
call_mgmt.xpm	msg_db16.xpm
call_mgmt16.xpm	out_dial.xpm
clock.xpm	out_dial16.xpm
clock16.xpm	prompt.xpm
comm.xpm	prompt16.xpm
comm16.xpm	sql.xpm
cucb.xpm	sql16.xpm

db.xpm	user.xpm
db16.xpm	user16.xpm
execute.xpm	voice.xpm
execute16.xpm	voice16.xpm
fax.xpm	voicetek.xpm
fax16.xpm	

Cell builder icons

Any icons associated with a customized cell that you developed with the Cell Builder must reside in the `ui_lang` file.

Control panel button label icons

The icons in Table B-4 are used in the Control Panel as button labels. All icons are encoded in XPM (X Pixmap) format. The graphical Symposium OPEN IVR Control Panel supports color and monochrome systems. When the Control Panel is launched, it tries to load icon files that end in `_4.xpm` (4-bit or 16-color icons). If this fails, it loads the set ending in `_1.xpm` (which contains only black and white icons).

Table B-4: Control Panel icons

uif_database_1.xpm	uif_lock_4.xpm
uif_database_4.xpm	uif_logo_1.xpm
uif_editor_1.xpm	uif_logo_4.xpm
uif_editor_4.xpm	uif_prompt_1.xpm
uif_help_1.xpm	uif_prompt_4.xpm
uif_help_4.xpm	uif_reports_1.xpm
uif_icon_1.xpm	uif_reports_4.xpm
uif_icon_4.xpm	uif_service_1.xpm
uif_lock_1.xpm	uif_service_4.xpm

Help files

When you select the option Help from a menu or a cell in the Application Editor, the system displays the appropriate cell Help file.

> ***Note:*** Help files for customized cells created with Cell Builder must be placed in the `ui_lang` file. Refer to Release 4.0 of the *Symposium OPEN IVR Application Development Guide* and the *Symposium OPEN IVR Cell Catalog* for additional information on customized cells.

Error log

Many OPEN IVR processes log events to the error log. The files in Table B-5 contain the text that appears in log entries. Previous versions of OPEN IVR maintained these files in the `log.d` directory.

Table B-5: Error log entries

adsi_erl.str	qds_erl.str	trsl_erl.str
aud_erl.str	sad_erl.str	ueh_erl.str
cli_erl.str	sai_erl.str	usr_erl.str
cli_s_erl.str	sam_erl.str	ust_erl.str
csc_erl.str	sase_erl.str	vft_erl.str
dbs_erl.str	sde_erl.str	vip_erl.str
fds_erl.str	sds_erl.str	vrm_erl.str
gnetconfig_error.str	sys_erl.str	vtk_erl.str
ids_erl.str	trapGen_erl.str	vxcc_erl.str
ods_erl.str	trs_a_erl.str	xai_erl.str
plf_erl.str	trs_erl.str	xpm_erl.str
pmg_erl.str	trsc_erl.str	

Language files

Language files contain any text strings displayed within the OPEN IVR user interfaces. You can edit virtually everything that appears on the screen or in menus by modifying these files.

Language files contain a table of token (or menu option) and string pairs.



CAUTION

Risk of system corruption

Do not alter the Language menu options listed in Table B-6.

DO NOT MODIFY THEM.

Ensure that the string is on the same line as the menu option and separated from it by one or more spaces or tabs.

Enclose the string in double quotes. The string may consist of any sequence of printable ASCII characters or spaces. To designate a line as a comment, start it with a pound sign (#). For clarity, you can use both comments and blank lines.

> ***Note:*** Pay close attention to any comments within the files, noting limitations on string lengths. It is possible to insert strings that are too long to fit in the space provided on the screen.

Make copies of these files before modifying them. In addition, if a string contains formatting characters (%s, %d, etc.), ensure that these are not removed from the string or reordered, or unpredictable behavior may result.

The language files, named for their corresponding processes, follow in Table B-6.

Table B-6: Language files with corresponding processes

Language file	Corresponding process
cli.lng	CLI (Cell Logic Interpreter)
cucb.lng	CUCB (Customizable User Cell Builder)
gmessage.lng	Gmessage (Generation Message Tool)
gnetconfig.lng	Gnetconfig (Graphical Network Configuration)
plf.lng	PLF (Prompt Loading Facility)
sai.lng	SAI (System Administration Interface)
scm.lng	SCM (System Configuration Manager)
sde.lng	SDE (System Database Editor)
uif.lng	UIF (User Interface)
xae.lng	XAE (Graphical Application Monitor)
xai.lng	XAI (Graphical System Monitor)
xbup.lng	XBUP (Graphical Backup Tool)
xconsole.lng	Xconsole (Graphical Control Panel)
xpm.lng	XPM (Graphical Prompt Manager)

SmartHelp

The `uif.hlp` file contains the SmartHelp (context-sensitive help) text used for every menu selection in the control panel interface.



CAUTION

Risk of system corruption

The files in Table B-7 are used internally by the Control Panel process.

DO NOT MODIFY THEM.

Table B-7: Control Panel process files

uif_lng.uid	Motif user interface file
uif_menu.uid	Motif user interface file
uif_login.uid	Motif user interface file

Modifying the uifg.hlp file

When the system reads the `uif.hlp` file, it associates sections of text with pull-down menu selections. Sections of Help text appear in the SmartHelp window when the user enters a menu selection with the mouse button depressed.

If you modify the pull-down menu option strings in the `uif.lng` file, be sure to modify the text enclosed within brackets in the `uif.hlp` file as well. The text in these files must match, or context-sensitive Help will not function.

Multiple speech recognition vocabularies

This feature allows you to easily internationalize the speech-recognition component of your OPEN IVR applications. With minimal modification to existing applications, you can have your OPEN IVR application recognize caller voice input in any language installed on the MRS.

This feature works through a combination of LANG and VINP cells. You set the language at the beginning of the application by incorporating a LANG cell for each language you plan to use. By setting a language in the LANG cell and then setting the VINP cell's "Use Common Vocabularies" parameter, the VINP cell can recognize spoken input in any language in use by the application. This allows you to develop applications independently of their intended language or languages. It also lets you easily retrofit existing speech-recognition applications with international speech-recognition ability.

This feature provides flexibility beyond internationalization. By installing the appropriate vocabularies, you can adapt this feature to virtually any speech-recognition application.

Languages and vocabularies

Each language has vocabulary subgroups. The vocabularies contain the actual words OPEN IVR recognizes. Table B-8 illustrates the relationship between languages and vocabularies.

Table B-8: Languages with subgroups

Language	Vocabulary	Contents
English	Digits	oh, zero, one, two,..., nine
English	Yes, No	yes, no
English	Months	January, February,..., December
English	Colors	Red, Orange,..., Violet
French	Digits	zero, un, deux,...,neuf
French	Yes, No	oui, non
Spanish	Digits	uno, dos,..., nueve
Spanish	Yes, No	si, no
Spanish	Months	Enero, Febrero,.., Diciembre

From this table, you can organize vocabularies into two different groups: Common and Unique.

Common vocabularies are vocabularies shared by all languages. A list of common vocabularies from the previous examples include the following, since all three languages share these vocabularies:

- Digits
- Yes, No

Unique vocabularies are vocabularies used by a single language or a group of languages, but are not shared with all the languages. A list of unique vocabularies from the previous example include the following:

- English Months
- Spanish Months
- English Colors

Internationalization through the LANG and VINP cells

The first part of internationalizing speech recognition consists of placing a VINP cell in your application at the point where you expect voice input. By selecting “Yes” in the VINP cell’s “Use Common Vocabularies” parameter, you activate the Common Vocabularies field. Clicking the browser in this field brings up a list of all of the common vocabularies on the MRS. By selecting a common vocabulary, you can have speech recognition in any language installed on the MRS.

The second part of internationalized speech recognition consists of adding a LANG cell for each language you have installed on your MRS. You must place the LANG cells before the VINP cells.

In each LANG cell, set the Language parameter to a language installed on the MRS. Figure B-1 illustrates the LANG cell parameters for a LANG cell set for French. Figure B-2 shows a LANG cell set for Spanish. Figure B-3 shows a VINP cell set to use the DIGITS common vocabulary.

Note: When configuring your MRS for speech recognition, keep the following in mind: OPEN IVR uses speech recognition for the express purpose of DTMF replacement. Therefore, when using the MRS software to assign return values to vocabulary words, use only the values 0 — 15. Since there are 16 keys on a DTMF keypad (0 — 9, *, #, A-D), OPEN IVR recognizes only 0 — 15 (16 digits) as valid return codes.

Table B-9 shows the relationship between the DTMF keys on the telephone keypad and the value sent to the application server by the MRS.

Table B-9: DTMF/ASCII relationship

DTMF digit	ASCII character sent
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
0	A
*	B
#	C
A	D
B	E
C	F
D	0

Figure B-1: LANG cell configured for French

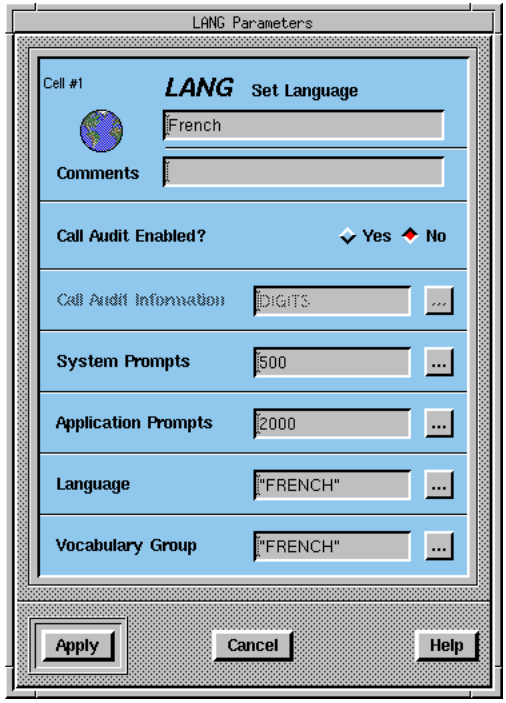
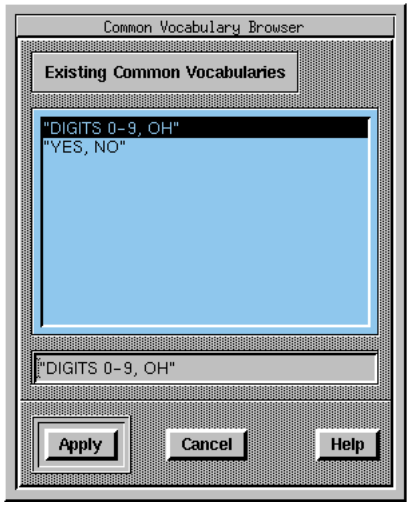


Figure B-2: LANG cell configured for Spanish



Figure B-3: VINP common vocabulary selected



Creating the mvrv.config file

The Multiple Voice Response Vocabularies feature requires the creation of the `mvrv.config` file. The `mvrv.config` file must reside in the OPEN IVR `user_data` directory. This file contains a list of all languages and vocabularies installed on the MRS.

To create the `mvrv.config` file, you must obtain a list of the languages and vocabularies installed on the MRS (see the *Symposium OPEN IVR Service Console Interface Reference Manual*). Enter the languages and vocabularies on this list into the `mvrv.config` file so OPEN IVR knows which languages and vocabularies it has access to.

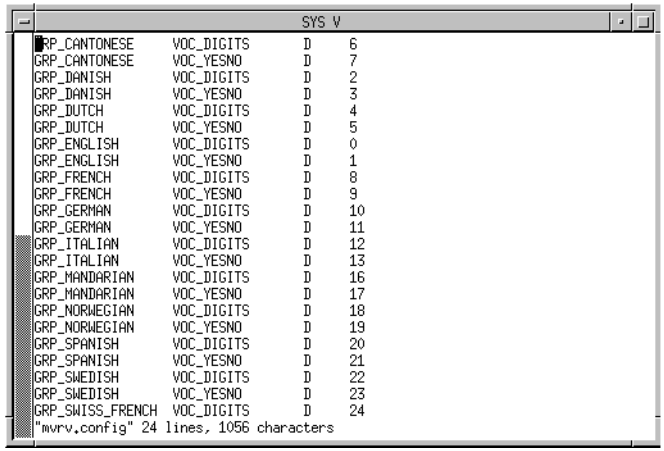
The `mvrv.config` file is made up of the following four columns:

- First column — lists the language group token from the `xae.lng` file located in a subdirectory specified by the environment variable `VOICE_LANG` off the `ui_lang` directory. This tells OPEN IVR which languages are available. To enter a language group token in this column, enter the language token as it appears in the `xae.lng` file.

- Second column — lists the vocabulary group token from the `xae.lng` file. This tells OPEN IVR which vocabularies are available. To enter a vocabulary group token in this column, enter it as it appears in the `xae.lng` file.
- Third column — lists the type of speech recognition used with the vocabulary. Enter C for Continuous or D for Discrete. This information is obtained from the list of languages installed on the MRS.
- Fourth column — lists the vocabulary's MRS Identification Number. This number is included on the list of languages installed on the MRS.

Figure B-4 shows an example of the `mvrv.config` file.

Figure B-4: mvrv.config file


SYS V
GRP_CANTONESE VOC_DIGITS D 6
GRP_CANTONESE VOC_YESNO D 7
GRP_DANISH VOC_DIGITS D 2
GRP_DANISH VOC_YESNO D 3
GRP_DUTCH VOC_DIGITS D 4
GRP_DUTCH VOC_YESNO D 5
GRP_ENGLISH VOC_DIGITS D 0
GRP_ENGLISH VOC_YESNO D 1
GRP_FRENCH VOC_DIGITS D 8
GRP_FRENCH VOC_YESNO D 9
GRP_GERMAN VOC_DIGITS D 10
GRP_GERMAN VOC_YESNO D 11
GRP_ITALIAN VOC_DIGITS D 12
GRP_ITALIAN VOC_YESNO D 13
GRP_MANDARIAN VOC_DIGITS D 16
GRP_MANDARIAN VOC_YESNO D 17
GRP_NORWEGIAN VOC_DIGITS D 18
GRP_NORWEGIAN VOC_YESNO D 19
GRP_SPANISH VOC_DIGITS D 20
GRP_SPANISH VOC_YESNO D 21
GRP_SWEDISH VOC_DIGITS D 22
GRP_SWEDISH VOC_YESNO D 23
GRP_SWISS_FRENCH VOC_DIGITS D 24
"mvrv.config" 24 lines, 1056 characters

Adding additional language and vocabulary groups

You can add either a language group token or a vocabulary group token to the `xae.lng` file by opening the `xae.lng` token file in an editor session and adding a line with two columns:

- First column — lists the language group token.
- Second column — lists the string to display in the editor.

For example, using the `xae.lng` file:

```
xae.lng (USA)en_US
      .
      .
      .
GRP_ENGLISH“ENGLISH”
VOC_DIGITS“DIGITS 0-9, OH”
```

Note: When editing the `xae.lng` file, be sure to press <Return> at the end of each line, including the last line.

Glossary

This glossary provides definitions of many terms used in the documentation for Symposium OPEN IVR. Some of these terms may have additional meanings not described here.

acoustic coupler

A device (such as an Audio Interface Unit) used to interface between sound equipment and a port on the MRS/AP.

Administration Tools

A set of tools, accessible from the Symposium OPEN IVR Control Panel, used to configure and control the Symposium OPEN IVR system.

ADSI

Analog Display Services Interface. The Bellcore standard defining the protocol used for the flow of information between a piece of telephone equipment (such as a switch, server, or voice mail) and a subscriber's telephone, PC, data terminal, or other device with a screen.

Analog Display Services Interface

See ADSI.

ANI

Automatic Number Identification. A protocol implemented in some trunk cards that records the calling number.

ANSI

American National Standards Institute.

AP

Application Processor. A computer or workstation running the AP software in a distributed configuration. Symposium OPEN IVR combines one or more Multimedia Resource Servers (MRSs) with a general-purpose Application Processor (AP) to provide a versatile application development and run-time environment. The AP provides the environment for developing, managing, and controlling the IVR applications.

AP software

Nortel's premiere voice application development and management software package that runs on the Application Processor (AP).

API

Application Programming Interface. Software libraries that enable the development of applications in a new environment, which communicate with applications in the legacy system. Software libraries that provide a standard access to services provided by a software package.

application

1 In Symposium OPEN IVR, a customized program that controls activity on one or more telephone trunks connected to an MRS voice system.

2 On a host computer, any type of program that carries out a task.

application developer

A person who creates Symposium OPEN IVR applications.

Application Editor

A utility used to graphically create or edit interactive voice response applications.

application processor

See AP.

Application Programming Interface

See API.

Automatic Number Identification

See ANI.

branch

A pathway from one cell in an application to the next cell.

buffers

A part of computer memory used to store information from one cell and pass it to another cell.

call flow

A diagram of a Symposium OPEN IVR application.

cell

The basic element of an application. Each cell causes some action to be performed—for example, playing a prompt to a caller. Each cell has a set of branches to other cells. After the cell performs its action, it determines which branch the application should follow to the next cell.

channel

1 A telephone trunk within a cluster of MRSs.

2 A path of communication between the AP and a communications or transmission facility. Channel sequences can span multiple MRSs, and their numbering starts at zero (0). Also referred to as a **port**.

CO

Central Office.

Control Panel

The Symposium OPEN IVR panel containing the buttons from which you can access the Administration, Information Database Editor, Report, Application, Prompt Management, and SmartHelp tools.

Dialed Number Identification Service

See DNIS.

DID

Direct inward dialing. A protocol used by some telephone equipment that allows callers to dial directly to an extension within a telephone system without going through a switchboard.

direct inward dialing

See DID.

DNIS

Dialed Number Identification Service. A feature of 800 and 900 numbers, in which the number dialed or a portion of it is sent to the MRS and stored in the DIGITS buffer in OPEN IVR.

DTMF

Dual Tone Multi-Frequency. Touchtone (push-button) dialing. DTMF is used to encode digits over analog telephone lines. Applications can collect information from callers by having them press telephone keys to create DTMF tones.

DTMF only mode

The default data input mode used by a Symposium OPEN IVR application to process caller input.

Dual Tone Multi-Frequency

See DTMF.

EXEC

The cell that executes an OPEN IVR application on the current channel.

falsing

A condition in which certain frequencies in a voice recording are incorrectly interpreted as DTMF tones. In some applications, false DTMF tones may cause the system to perform unwanted activities, such as selecting menu choices, terminating playback, or hanging up prematurely.

field

The specific location of information (data) within a record in a relational database.

GDAT

Play Prompts and Get Data. A cell that allows you to play one or more prompts and collect up to 16 digits of input from the caller.

ground start

A method of signaling on the trunk in which one side of the two wires (typically the ring wire) is momentarily grounded to obtain a dial tone. This type of signal is normally used with PBXs.

hook-flash

Depressing and releasing the plunger or handset-cradle on a telephone.

Information Database Editor

A utility used to create a message or information database, and to edit an information database.

IVR

Interactive Voice Response.

IVR Developer

A licensed component of AP software that provides application development tools to graphically create or edit interactive voice response applications.

link

A physical connection between two devices.

MF tones

Multifrequency tones. These are tones used by ANI. Symposium OPEN IVR applications can receive and store MF digits when the ANI option is in use.

MRS

Multimedia Resource Server. Symposium OPEN IVR combines one or more Multimedia Resource Servers (MRSs) with a general-purpose Application Processor (AP) to provide a versatile application development and/or run-time environment. The MRS provides the telephony, information processing (voice, fax, speech recognition, and text-to-speech) environment.

MRS/AP

A single unit that runs a “folded” configuration; that is, a configuration that includes both the Multimedia Resource Server and the Application Processor. The MRS/AP contains the functionality of the other platforms, but in a single hardware unit.

MRS/Rack

A unit comprising a rack-mountable Application Processor (AP) and a rack-mountable Multimedia Resource Server (MRS). Multiple MRS/Racks can be installed in cabinets and stacked to take up less room.

MRS/Tower

The MRS/Tower platform is comprised of an Application Processor and a Tower Multimedia Resource Server.

multifrequency tones

See MF tones.

Multimedia Resource Server

See MRS.

node

A grouping composed of an AP (running Symposium OPEN IVR) connected to one or more MRSs.

parameter

A variable used to define the function of Symposium OPEN IVR cells. You set parameters in the parameter window for each cell (when using the Application Editor), or in the cell window (when using the character-based Application Editor).

PBX

Private Branch Exchange. A private phone system that allows communication within a business and between the business and the public telephone system.

PDAT

Play Prompts with Data. A cell that allows you to play prompts to announce the contents of a buffer. The buffer content is read according to the specified type.

Pencil mode

Refers to the period during which the branch connect tool is active, and the pencil cursor is displayed. During pencil mode you can select multiple cell connectors.

prompt

A voice recording that helps lead a caller through a Symposium OPEN IVR application.

Prompt Management tools

A set of tools, accessed from the Symposium OPEN IVR Control Panel, that is used to record, load, and update prompts.

record

1 To store sound on tape or disk for later playback.

2 In a database, a group of related data items (fields) treated as one item.

Reorder tone

A reorder tone is a fast busy or channel busy tone, which is applied 120 times per minute. The tone means that all switching paths are busy, all toll trunks are busy, there are equipment blockages, the caller dialed an unassigned code, or the digits dialed were scrambled.

Reports tools

A set of tools, accessed from the Symposium OPEN IVR Control Panel, that is used to generate reports about the system's activities.

socket

An end point (portal) from one TCP/IP application to another.

subscriber

A person serviced by a voice processing application (such as voice mail).

Telco

Telephone company.

TRS

Terminal Resource Server. The Symposium OPEN IVR process that manages the assignment of system resources to the Host Link products.

VINP cell

An OPEN IVR cell which toggles data input mode between voice input and DTMF only (default) mode.

voice input mode

When enabled by the VINP cell, this mode allows the OPEN IVR application to accept both voice and DTMF input.

wink

A momentary interruption in single frequency tone that indicates the distant central office (CO) is ready to receive the digits that were just dialed.

X-Window Application Editor

See XAE.

XAE

X-Window Application Editor. Another term for the Application Editor.

Index

A

abort digit, 3-17
abort prompt, 3-17
accessing
  AP software, 1-3
  record numbers, 7-17
  saved information from calls, 4-9
  specific record number, 7-18
  User Function from the call flow, 8-3
adding
  language groups, B-19
  messages, 6-8
  vocabulary groups, B-19
administration tools, 1-4
ADSI
  application, designing an, 4-22
  cells, 4-24
  telephone, 4-21
advanced User Function techniques, 8-25
alternate method, controlling OPEN IVR execution after start-up, 1-18
AMSG cell, 6-3
  customizing, 5-18
Analog Display Services Interface (ADSI), 4-20
ANSW cell (cell 1)
  customizing, 5-14
  placing and naming, 5-13
AP software
  accessing, 1-3
  auto-restart feature, 1-16
application
  basics, 4-2
  call flow, collect call, 4-18
  cell branches, 3-2
  components, application, 4-1
  constructing an, 4-27
  controlling execution, 5-27
  debugging an, 5-23, 5-25
  default fields, 3-13
  defaults, establishing, 3-11
  design an, 4-28
  design, concepts, 4-1
  designing, 5-2
  editing, 4-29, 5-22
  editor button label icons, B-6
  example, 4-4
  example of an, 4-3
  for leaving a message, 6-10
  for retrieving messages, 6-10
  install and run, 4-29
  LeaveMsg, 6-15
  loading and running, 5-22
  MakeMailbx, 6-11
  MsgMenu, 6-13
  presenting a menu, 6-10
  RetrievMsg, 6-18
  reviewing, 5-21
  saving, 5-21
  to create mailboxes, 6-10
  tools, 1-6
  using the COUT and CEND cells, 4-17
  with branching, example, 4-5
Application Editor
  Edit menu
    Paste, 2-30
    Remove, 2-32

menu bar, 5-23
menu bar in Debug mode, 5-23
opening, 4-29
selecting the destination cell, 2-29
using, 5-8
View menu, unconnected cells, 2-21
Application File
creating, 4-29
Application Programming Interface (API) for issuing level 3 ADSI commands, 4-27
applications
about cells, 3-1
and the current message, 6-4
branching, 4-1
building, 4-28
error handling, 8-7
sample, building, 5-1
applying
new default parameters, 3-26
ASCII file to a database, importing information from, 7-34
audited information, logging, 4-33
Automatic Speech Recognition
LISTEN cell, 4-30
auto-restart feature, AP software, 1-16

B

back to beginning digit, 3-20
background menu, sample, 1-16
backtrack digit, 3-19
backtrack/skip ahead time, 3-22
branches, 3-2, 4-4
building applications, 4-1
Breakpoint Information window, 5-29
breakpoints
deleting, 5-30
displaying, 5-29
setting, 5-28
buffer labels dialog box, 8-35
buffers
displaying, 5-32
input and output, 8-6
modifying, 5-31, 5-33
overview, 4-6
system, 4-7
user-defined, 4-12
building
applications, 4-28
sample application, 5-1
User Function, 8-11
button label icons, application editor, B-6
buttons, sliders, and entry boxes, 1-11

C

call flow
for LEAVEMSG, 5-5
MakeMailbx application, 6-12
MsgMenu application, 6-14
top-to-bottom, 4-3
Cell Builder, 1-6, 8-30
creating cells, 8-36
saving cells, 8-36
specifying tokens, 8-38
Cell Palette
identifying group, 8-40
cells
about, 3-1
ADSI (ds), 4-24
application using the COUT and CEND, 4-17
branches, 3-2
cell builder icons, B-7
cell group, 3-2
cell icon, 3-1
Cleanup Handler, 3-26
components, 3-1
customizing, 4-29, 5-14, 5-17
DFLT, 3-12
hang, 4-3
name, 3-2
naming, 5-12, 5-13
number, 3-2
off-page connectors, 2-28
Parameters page, 3-3, 3-4

parameters, START, 3-7
placing, 5-12, 5-13
selecting, 4-29
skip to end digit, 3-19
speedup digit, 3-20
START, 3-5
STOR, 5-13
types, message database, 6-3
unconnected, 4-4
USER Cell Builder, 8-30
user-defined name, 3-2
See also cells and off-page connectors, 2-28
changing
directories, 5-9
or inserting data in records, 7-16
START cell Message Database Name parameter, 5-11
channel
debugging, 5-26, 5-27
CHEK cell, 7-22
choosing
database type, 6-8, 7-10
number of significant digits, 6-6
Cleanup handler cell, establishing, 3-26
CMSG (Copy Message), 6-3
coding
function, 8-3
reply, 8-4, 8-6
User Function, 8-10
collect call application call flow, 4-18
command-line messages, viewing, 7-37
components of a cell, 3-1
concepts of application design, 4-1
confidence levels
evaluating, 4-31
connecting, the next cell, 5-15
console
accessing, 1-3
system, 1-3
constructing an application, 4-27
control execution of an application using the options Interrupt and Continue to, 5-30
Control Panel, 1-1
exiting, 1-15
control panel, 1-2
button label icons, B-7
pull-down menus, 1-9
restoring, 1-15
controlling
execution of an application, 5-27, 5-31
OPEN IVR execution after start-up, 1-17
alternate method, 1-18
preferred method, 1-17
creating
an Application File, 4-29
application with the Application Editor, 5-8
cells, 8-36
database, 6-6, 7-33
database template, 7-12
development directory, 4-30
help file for the new cell type, 8-43
information database, 7-7
mailboxes, 6-8, 6-10
mvrv.config file, B-18
records, 7-13
STOR cell, 5-13
User Functions, 8-1
current message, applications and, 6-4
cursor keys, 1-12
customizable
user cell builder
buffer labels, 8-35
customization and internationalization, B-1
customizing
AMSG cell (cell 5), 5-18
ANSW cell (cell 1), 5-14
cells, 5-14, 5-17
DMSG cell (cell 10), 5-20
DMSG cell (cell 12), 5-20
first MENU cell (cell 6), 5-18
HANG cell (cell 14), 5-20
PLAY cell (cell 11), 5-20
PMSG cell (cell 8), 5-19
RMSG Cell (Cell 3), 5-16

second MENU cell (cell 9), 5-19
STOR cell (cell 2), 5-15
user cell, 8-35
USER cells, 8-29
customizing cells, 4-29

D

data
exporting, 7-31
importing, 7-32
database, 4-12
creating a, 7-33
creating and naming, 6-6
editor, information, 1-5
example, 6-9
extracting information, 7-22
information, 4-12, 4-14
exporting to an ASCII file, 7-34
using through the application, 7-20
loading, 7-19
locating information, 7-22
message, 3-14, 4-13
message database, 4-12
naming, 7-9
records, determining information, 7-6
retrieving messages, 6-9
SQL, 3-24
template, creating, 7-12
templates, 7-11
type, choosing, 6-8, 7-10
types, 3-24
databases
creating mailboxes, 6-10
DDEL cell, 7-29
debug and test, 4-29
Debug mode, Application Editor menu bar, 5-23
debugging
an application, 5-23, 5-25
overview, 5-23
application, 5-27
channel, 5-26
default cell Get Data Termination touch-tone keypad window, 3-10
default fields, application, 3-13
deleting
breakpoints, 5-30
record numbers, 7-19
design an application, 4-28
designing
ADSI application, 4-22
application, 5-2
User Function, 8-9
determining information for database records, 7-6
developing a sample application, 5-1
development directory, creating, 4-30
DFLT cell, 3-12
DFLT cell parameters, modifyingcells
modifying parameters, 3-5
dialog box
buffer labels, 8-35
digit
abort, 3-17
back to beginning, 3-20
backtrack, 3-19
pause, 3-14
resume, 3-15
skip ahead, 3-18
skip to end, 3-19
slowdown, 3-21
speedup, 3-20
termination, 3-18
digits
number of significant digits, 6-6
DINS cell, 7-27
directing rejected responses, 4-32
directories, changing, 5-9
displaying
breakpoints, 5-29
buffers, 5-32
DMSG
(Delete Message), 6-4
cell (cell 10), customizing, 5-20
cell (cell 12), customizing, 5-20

Drawing Board tool
description, 2-3
See also tools, 2-3
dsCON—Connect Session, 4-25
dsDATA—data input forms, 4-26
dsDIS—disconnect session, 4-26
dsINFO—display information, 4-27
dsINPUT—get data, 4-26
dsKEYS—define soft keys, 4-26
dsMENU—display menu, 4-26
dsMODE—choose voice or data mode, 4-27
dsUSER—connect ADSI user function, 4-27
DTMF functionality, supporting, 4-31

E

Edit menu, choosing the Add options, 2-34
editing
application, 4-29, 5-22
records, 7-15
Start cell, 4-29
STARTcell, 5-10
templates, 7-15
enabling Feature Download, 4-26
entry boxes
buttons, sliders, and, 1-11
text, 1-11
error handling, 8-7
error log, B-8
establishing
application defaults, 3-11
Cleanup Handler cell, 3-26
evaluating multiple results with associated confidence levels, 4-31
example
an application using buffers, 4-11
application, 4-3
applications using message database, 6-9
mailbox in a message database, 6-2
outdialing application, 4-16
simple application, 4-4
simple application with branching, 4-5
exceptions, reporting, 4-33
execution of an application, controlling, 5-27
exiting
from Control Panel, 1-15
from the Information Database Editor, 7-19
exporting
and importing data, formats, 7-30
data, 7-31
database information to an ASCII file, 7-34
extracting
data, 7-22

F

Feature Download, enabling, 4-26
file, source, 8-6
first MENU cell (cell 6), customizing, 5-18
formats for exporting and importing data, 7-30
formatting returned results, 4-32
function code, 8-3
functions, user-defined, 8-8

G

Get Data Termination, 3-21
touch-tone keypad window, default cell, 3-10
GMSG (Get a Message), 6-3
graphical
debugger menu options, 5-24
user environment, 1-10
Graphical Debugger
invoking, 5-25
quitting, 5-34
greetings
playing back, 7-20
group, cell, 3-2

H

handling, error, 8-7
HANG cell (cell 14), customizing, 5-20
hang cells, 4-3

help files, B-8
host name, 3-25

I

icons
  cell builder, B-7
  control panel button label, B-7
identification numbers for prompts and messages, 4-15
identifyingLgroup, 8-40
importing
  and exporting data, formats, 7-30
  data, 7-32
  information from an ASCII file to a database, 7-34
including the User Function in an application, 8-14
information database
  creating an, 7-7
  editor, 1-5
  loading, 7-19
  locating and extracting data, 7-22
  planning an, 7-3
  records in an, 4-15
Information Database Editor
  exiting, 7-19
  opening, 6-5, 7-7
information databases, 4-14
input buffers, 8-6
inserting or changing data in records, 7-16
install and run an application, 4-29
interdigit timeout, 3-23
interface language directory, user, B-2
internationalization
  customization and, B-1
  through the LANG and VINP cells, B-13
invoking the Graphical Debugger, 5-25
items in the UI_LANG directory, B-6

L

LANG and VINP cells, internationalization through, B-13
language
  and vocabulary groups, adding, B-19
  files, B-9
languages and vocabularies, B-12
LEAVEMSG
  application, 5-3
  call flow for, 5-5
  prompt number descriptions, 5-8
LEAVEMSG application
  cells, 5-14
LeaveMsg application, 6-15
leaving a message, an application for leaving, 6-10
level 3 ADSI commands, Application Programming Interface (API) for issuing, 4-27
LISTEN cell
  Automatic Speech Recognition, 4-30
  confidence level, 4-31
loading
  and running the application, 5-22
  information database, 7-19
locating data in an information database, 7-22
location, storage, 7-5
locking the screen, 1-13
logging audited information, 4-33

M

mailboxes
  creating, 6-8
  in a message database, 4-14
  in message database, example, 6-2
MakeMailbx application, 6-11
  call flow, 6-12
management, tools, prompt, 1-7
Maximum Pause Time, 3-21

menu
application for, 6-10
option requiring verification, 1-12
options, graphical debugger, 5-24
menu bar, Application Editor, 5-23
message database
creating mailboxes, 6-10
mailboxes in a, 4-14
retrieving messages, 6-9
message databases, 3-14, 4-12, 4-13
adding messages
databases
adding messages, 6-8
cell types, 6-3
examples, 6-9
leaving a message, 6-10
MakeMailbx, 6-11
password, 6-3
presenting a menu, 6-10
using, 6-1, 6-8
messages
adding, 6-8
identification numbers, 4-15
retrieving, 6-9
Minimum Voice Duration, 3-23
modifying
buffers, 5-31, 5-33
DFLT cell parameters, 3-5
resource files and colors, B-1
uifg.hlp file, B-11
MsgMenu application, 6-13
call flow, 6-14
MSGS (Get Message List), 6-3
multiple speech recognition vocabularies, B-11
mvrv.config file, creating, B-18

N

name
cell, 3-2
user-defined, 3-2
naming
cells, 5-12, 5-13
database, 6-6
the database, 7-9
nedit editor, using to create multiple records, 7-36
next cell, connecting, 5-15
number
cell, 3-2
of significant digits
specifying, 7-10

O

off-page connectors, description, 2-28
OPEN IVR
execution after start-up, controlling, 1-17
opening
Application Editor, 4-29
Information Database Editor, 6-5, 7-7
outdialing, 4-16
application, example, 4-16
output buffers, 8-6
overview
buffers, 4-6
debugging an application, 5-23

P

panel, control, 1-2
parameters
default, 3-26
Parameters page, 3-3, 3-4
password, 3-25, 6-3
pause
digit, 3-14
prompt, 3-15
timeout action, 3-22
performing database functions from the UNIX command line, 7-33

placing
cells, 5-12, 5-13
ANSW cell (cell 1), 5-13
planning an information database, 7-3
PLAY cell (cell 11), customizing, 5-20
playing back specialized greetings, 7-20
PMSG
(Play a Message), 6-3
cell (cell 8), customizing, 5-19
potentially critical functions, safety features, 1-12
preferred method
controlling OPEN IVR execution after start-up, 1-17
process, User Function, 8-5
prompt
abort, 3-17
identification numbers, 4-15
management tools, 1-7
number descriptions, LEAVEMSG, 5-8
pause, 3-15
resume, 3-16
selecting, 4-29, 5-8
Prompt Loading Facility (PLF), prompts, recording, 1-7
pull-down menus, using, 1-9

Q

quitting, Graphical Debugger, 5-34

R

record layouts, understanding, 7-4
record numbers
accessing, 7-17
deleting, 7-19
records
creating, 7-13
editing, 7-15
in an information database, 4-15
rejected responses, directing, 4-32
reply code, 8-4, 8-6
report tools, 1-5
reporting exceptions, 4-33
resource files and colors, modifying, B-1
restoring the control panel, 1-15
resume
digit, 3-15
prompt, 3-16
retrieving messages, 6-9
an application for, 6-10
RetrievMsg application, 6-18
returned results, formatting, 4-32
reviewing the application, 5-21
RMSG
(Record Message), 6-3
Cell (Cell 3), customizing, 5-16
running
application, 5-27

S

safety features on potentially critical functions, 1-12
sample
application
building, 5-1
developing, 5-1
background menu, 1-16
saved information from calls, accessing, 4-9
saving
application, 5-21
cells, 8-36
screen
lock, 1-13
unlocking, 1-14
second MENU cell (cell 9), customizing, 5-19
selecting
cells, 4-29
prompts, 4-29
selecting, prompts, 5-8
setting breakpoints, 5-28
significant digits, choosing the number, 6-6
simple application

example, 4-4
with branching, example, 4-5
skip
ahead digit, 3-18
ahead/backtrack time, 3-22
to end digit, 3-19
slowdown digit, 3-21
SmartHelp, 1-8, B-11
source file, 8-6
specialized greetings, playing back, 7-20
specific
record number, accessing, 7-18
specifying
number of significant digits, 7-10
tokens in Cell Builder, 8-38
speech recognition, multiple vocabularies, B-11
speedup digit, 3-20
SQL
database name, 3-24
DBMS type, 3-24
maximum server count, 3-24
Standard 1.0, using, 7-1
standard prompts, A-1
START
cell
Message Database Name parameter, changing, 5-11
Pause Digit, 3-12
Resume Digit, 3-21
START cell, 3-5
editing, 5-10
parameters, 3-7
Pause Digit, 3-12
Start cell, editing, 4-29
Status window
displaying applications on the drawing board, 2-4
displaying cells on the drawing board, 2-4
STOR cell
creating, 5-13
naming, 5-13
STOR cell (cell 2)
customizing, 5-15
storage location, 7-5
supporting
DTMF functionality, 4-31
system
buffers, 4-7
console, 1-3

T

Task Manager, 8-25
telephone, ADSI, 4-21
templates
database, 7-11
editing, 7-15
termination digit, 3-18
test and debug, 4-29
testing
User Function, 8-12
text entry
boxes, 1-11
time stamp, 6-2
timeout, 8-28
interdigit, 3-23
tools
administration, 1-4
application, 1-6
report, 1-5
top-to-bottom call flow, 4-3

U

UI_LANG directory, items in, B-6
uifg.hlp file, modifying, B-11
UMSG (Update Message List), 6-4
unconnected cells, 4-4
understanding
record layouts, 7-4
User Function process, 8-2
UNIX command line
performing database functions from, 7-33
using SDE commands from the, 7-36
unlocking the screen, 1-14

UPDT cell, 7-25
UPWD (Update Password), 6-4
user
  cell, customizing, 8-35
  environment, graphical, 1-10
USER Cell Builder, 8-30
USER cell builder
  menu
    Close, 8-32
    Exit, 8-32
    New, 8-31
    Open, 8-31
    Overview, 8-32
    Save, 8-32
    Save as, 8-32
User Cell Builder
  menu items
    About, 8-33
USER cells
  custominzing, 8-29
User Function
  accessing from the call flow, 8-3
  advanced techniques, 8-25
  building, 8-11
  coding a, 8-10
  creating, 8-1
  designing a, 8-9
  including in an application, 8-14
  process, 8-5
    understanding, 8-2
  testing, 8-12
user interface language directory, B-2
User Name, 3-25
user-defined
  buffers, 4-12
  functions, 8-8
  name, 3-2
using
  database information through the application, 7-20
  menu options Step and Next Cell to control the execution of an application, 5-31
  message databases, 6-1, 6-8
  nedit editor to create multiple records, 7-36
  options Interrupt and Continue to control the execution of an application, 5-30
  pull-down menus, 1-9
  SDE commands from the UNIX command line, 7-36
  Standard 1.0, 7-1
usr.c file, 8-15

V

viewing command-line messages, 7-37
vocabularies
  multiple speech recognition, B-11
vocabularies and languages, B-12
Voice Duration, Minimum, 3-23

Symposium OPEN IVR

Application Development Guide

Toronto Information Products
Nortel
522 University Avenue, 14th Floor
Toronto, Ontario, Canada
M5G 1W7

© 1998 Northern Telecom
All rights reserved

Information is subject to change without notice. Northern Telecom reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

Symposium, IVR, and Nortel are registered trademarks of Northern Telecom in the U.S. and Canada. Meridian 1 is a registered trademark of Northern Telecom. DEC and VT 420 are registered trademarks of Digital Equipment Corporation. HP, LaserJet, and ThinkJet are registered trademarks of Hewlett-Packard Company. Motif is a trademark of Open Software Foundation Inc. UNIX is a registered trademark of Novell, Inc. X Window System and X are trademarks of the Massachusetts Institute of Technology.

Product release: 4.0
Document release: Standard 1.0
Date: July 1998

Printed in the United States of America

NORTEL
NORTHERN TELECOM